ESD-TR-86-278

MTR 10090

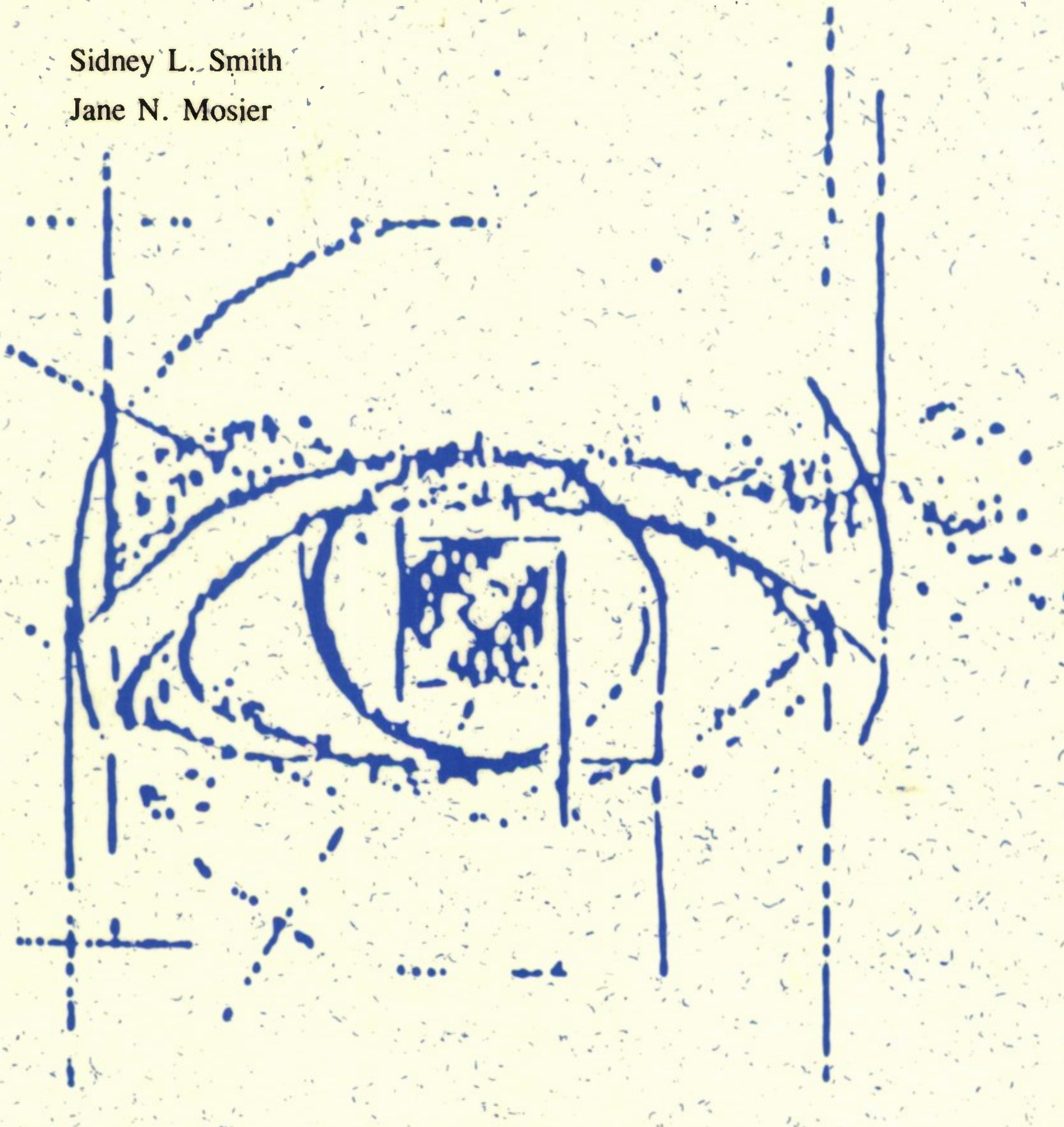
August 1986

ADA 177 198

Guidelines for Designing User Interface Software

Sidney L. Smith

Jane N. Mosier



Prepared for Deputy Commander for Development Plans and Support Systems, Electronic Systems Division, AFSC, United States Air Force, Hanscom Air Force Base, Massachusetts.

Approved for public release; distribution unlimited.

MITRE

Bedford, Massachusetts

REVIEW AND APPROVAL

This technical report has been reviewed and is approved for publication.

MICHAEL F. MERRIMAN, Lt, USAF
Project Officer
Requirements Analysis

ARTHUR G. DECELLES, Capt, USAF
Program Manager
Computer Resource Management
Technology Program

FOR THE COMMANDER

ROBERT J. KENT
Director
Software Design Center
Deputy Commander for Development Plans
and Support Systems

AD/A177 198

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) MTR-10090 ESD-TR-86-278	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION The MITRE Corporation 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) Burlington Road Bedford, MA 01730	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Deputy Commander for (cont.) 8b. OFFICE SYMBOL (If applicable) XRSE	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F19628-86-C-0001
8c. ADDRESS (City, State and ZIP Code) Electronic Systems Division, AFSC Hanscom AFB, MA 01731-5000	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO.: 64740F PROJECT NO.: 5220/5720 TASK NO.: WORK UNIT NO.:
11. TITLE (Include Security Classification) GUIDELINES FOR DESIGNING USER (continued)	
12. PERSONAL AUTHOR(S) Smith, Sidney L., Mosier, Jane N.	
13a. TYPE OF REPORT Final 13b. TIME COVERED FROM ______ TO ______	14. DATE OF REPORT (Yr., Mo., Day) 1986 August 01 15. PAGE COUNT 488
16. SUPPLEMENTARY NOTATION	

17. COSATI CODES		
FIELD	GROUP	SUB. GR.

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)

Computer-Based Systems
Design Guidelines
Human Factors
Information Systems
User-System Interface

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report proposes 944 guidelines for designing software to support the user interface to computer-based information systems. These design guidelines cover six functional areas: data entry, data display, sequence control, user guidance, data transmission, and data protection. This report revises and extends previously published guidelines, and is the final compilation under current Air Force sponsorship.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Diana F. Arimento	22b. TELEPHONE NUMBER (Include Area Code) (617)271-7454 22c. OFFICE SYMBOL Mail Stop D230

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

8a. Development Plans and Support Systems

11. INTERFACE SOFTWARE

SUMMARY

This report offers guidelines for design of user interface software in six functional areas: data entry, data display, sequence control, user guidance, data transmission, and data protection. This report revises and extends previous compilations of design guidelines (cf. Smith and Mosier, 1984a).

If you are a teacher, a student, a human factors practitioner or researcher, these guidelines can serve as a starting point for the development and application of expert knowledge. But that is not the primary objective of this compilation. The guidelines are proposed here as a potential tool for designers of user interface software.

If you are a system analyst, you can review these guidelines to establish design requirements. If you are a software designer, you can consult these guidelines to derive the specific design rules appropriate for your particular system application. That translation from general guidelines to specific rules will help focus attention on critical user interface design questions early in the design process.

If you are a manager responsible for user interface software design, you may find in these guidelines a means to make the design process more efficient. Guidelines can help establish rules for coordinating individual design contributions, can help to make design decisions just once rather than leaving them to be made over and over again by individual designers, can help to define detailed design requirements and to evaluate user interface software in comparison with those requirements.

The design of user interface software will often involve a considerable investment of time and effort. Design guidelines can help ensure the value of that investment.

ACKNOWLEDGMENT

This report was prepared by The MITRE Corporation. The work reported here was sponsored by the Directorate of Computer Systems Engineering, Deputy for Development Plans and Support Systems of the Electronic Systems Division (ESD) of the United States Air Force Systems Command, Hanscom Air Force Base, MA 01731. Continuing funding for this work was provided by the Air Force Computer Resource Management Technology Program, Program Element 64740F, under ESD/MITRE Project 5220. Final publication of these guidelines was funded under Project 5720.

The Computer Resource Management Technology Program supports development and transition into active use of tools and techniques needed to cope with the explosive growth in Air Force systems that use computer resources. The objectives of that Program are:

to provide for the transition to Air Force systems of computer system developments in laboratories, industry, and academia;

to develop and apply software acquisition management techniques to reduce life cycle costs;

to provide improved software design tools;

to address problems associated with computer security;

to develop advanced software engineering tools, techniques, and systems;

to support the implementation of high-order programming languages;

to improve human engineering of computer systems; and

to develop and apply computer simulation techniques in support of system acquisition.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Concluded)

INTRODUCTION

In designing computer-based information systems, special attention must be given to software supporting the user interface. For the past several years, guidelines for designing user interface software have been compiled as a continuing effort sponsored by the Air Force Electronic Systems Division (ESD). Five previous ESD reports have dealt with this subject (Smith, 1980; 1981a; 1982b; Smith and Aucella, 1983a; Smith and Mosier, 1984a).

This present report revises and expands previously published material, and proposes a comprehensive set of guidelines for design of user interface software in computer-based information systems. Although a great many changes have been made, much of the text and guidelines material in this report will seem familiar to the readers of previous reports.

Different people will read this report for different reasons — teachers and students, human factors practitioners and researchers, system analysts and software designers, and their managers. Each reader will bring to the task a unique background of experience and interests. Thus some introductory comments are needed to help familiarize readers with the general problems of user interface design and the particular need for guidelines to design user interface software.

For the skeptical reader, this introduction offers arguments in favor of guidelines for user interface software design. For the enthusiast who may imagine that guidelines can solve all design problems, this introduction will note some of their limitations. For those readers who wish to apply design guidelines, this introduction describes how the report is formatted, how the guidelines are presented and annotated, and concludes with some recommendations for how the guidelines should be used.

Information Systems

Computers today are used for a broad range of applications. User interface design guidelines cannot be applied usefully in every case. Some computers may be embedded as components in larger systems, so that they communicate only with other computers and not directly with human users. When there is no user interface, then no user interface design guidelines are needed.

Some computers are designed as general tools which can be adapted by skilled users for whatever purpose they desire. The particular tasks for which a general-purpose computer might be used are not defined in advance by the designer. Instead, a user must provide exact instructions to program the computer to perform any task at hand. The designer may try to ensure that the computer can process appropriate programming languages, but otherwise is not concerned with explicit design of a user interface.

Other computer systems are designed to help particular users perform specific tasks. Such computer applications are referred to here as information systems. Applications of information systems range from relatively simple data entry and retrieval (e.g., airline reservations) through more complex monitoring and control tasks (inventory control, process control, air traffic control) to jobs requiring long-term analysis and planning. Military command, control and communication systems span that broad range of information system applications.

To the extent that information systems support human users performing defined tasks, careful design of the user-system interface will be needed to ensure effective system operation. The guidelines proposed in this report are intended to improve user interface design for such information systems.

Users of information systems interact with a computer in order to accomplish information handling tasks necessary to get their jobs done. They differ in ability, training and job experience. They may be keenly concerned with task performance, but may have little knowledge of (or interest in) the computers themselves. Design of the user-system interface must take account of those human factors.

User-System Interface

What is the user-system interface? In common usage, the phrase is broadly defined to include all aspects of system design that affect system use (Smith, 1982a). This report, however, is concerned more narrowly with the user interface to computer-based information systems, i.e., with those aspects of system design that influence a user's participation in information handling tasks.

This report focuses even more narrowly on those design features of the user interface that are implemented via software (i.e., the design of computer program logic) rather than hardware (the design of equipment). The guidelines proposed here are generally worded in terms of the functions that a user must perform, and the functional capabilities that a designer should provide, rather than the particular physical devices that might be used to implement those functions. Thus a particular guideline might deal with "pointing" as a function, with no necessary recommendation whether pointing should be accomplished via touch display or lightpen or any other physical device.

It is obvious that software is not the only significant factor influencing user performance. Other aspects of user interface design are clearly important, including workstation design, physical display characteristics, keyboard layout, environmental factors such as illumination and noise, the design of paper forms and written documentation, user training courses, etc. To achieve a good user interface design, all of those factors must be designed with care. But designers must look elsewhere for advice on those topics. They are not covered in this report.

User Interface Software

The significant role of user interface software in system design poses a special challenge to human factors practitioners, recognized early by Parsons:

> . . . what sets data processing systems apart as a special breed? The function of each switch button, the functional arrangement among the buttons, the size and distribution of elements within a display are established not in the design of the equipment but in how the computer is programmed. Of even more consequence, the 'design' in the programs establishes the contents of processed data available to the operator and the visual relationships among the data. In combination with or in place of hardware, it can also establish the sequence of actions which the operator must use and the feedback to the operator concerning those actions.
>
> (1970, page 169)

Continuing concern for user interface software is suggested by phrases such as "software psychology" (cf. Shneiderman, 1980). But user interface design cannot be the concern only of the psychologist or the human factors specialist. It is a significant part of information system design that must engage the attention of system developers, designers, and ultimately system users as well. Those who look to the future of information systems predict that user interface design will become a specialty area; designers trained in both computer science and human factors will be employed to develop user interface software (Williges, 1984).

User interface software can represent a sizable investment of programming effort during initial system development, and later when a system is upgraded. In a recent survey (Smith and Mosier, 1984b), information system designers were asked to estimate the percent of operational software devoted to implementing the user interface. Overall, the average estimate was that user interface design comprises 30 to 35 percent of operational software. Estimates for individual systems ranged from 3 to 100 percent, reflecting the fact that some computer systems require a much higher investment in user interface design than others, depending upon their purpose.

Significance of the User Interface

The design of user interface software is not only expensive and time-consuming, but it is also critical for effective system performance. To be sure, users can sometimes compensate for poor design with extra effort. Probably no single user interface design flaw, in itself, will cause system failure. But there is a limit to how well users can adapt to a poorly designed interface. As one deficiency is added to another, the cumulative negative effects may eventually result in system failure, poor performance, and/or user complaints.

Outright system failure can be seen in systems that are underused where use is optional, or are abandoned entirely. There may be retention of (or reversion to) manual data handling procedures, with little use of automated capabilities. When a system fails in this way, the result is disrupted operation, wasted time, effort and money, and failure to achieve the potential benefits of automated information handling.

In a constrained environment, such as that of many military and commercial information systems, users may have little choice but to make do with whatever interface design is provided. There the symptoms of poor user interface design may appear in degraded performance. Frequent and/or serious errors in data handling may result from confusing user interface design. Tedious user procedures may slow data processing, resulting in longer queues at the checkout counter, the teller's window, the visa office, the truck dock, or any other workplace where the potential benefits of computer support are outweighed by an unintended increase in human effort.

In situations where degradation in system performance is not so easily measured, symptoms of poor user interface design may appear as user complaints. The system may be described as hard to learn, or clumsy, tiring and slow to use. The users' view of a system is conditioned chiefly by experience with its interface. If the user interface is unsatisfactory, the users' view of the system will be negative regardless of any niceties of internal computer processing.

A convincing demonstration of design improvement has been reported by Keister and Gallaway (1983). Those authors describe a data entry application in which relatively simple improvements to user interface software — including selection and formatting of displayed data, consistency in wording and procedures, on-line user guidance, explicit error messages, re-entry rather than overtyping for data change, elimination of abbreviations, etc. — resulted in significantly improved system performance. Data entry was accomplished 25 percent faster, and with 25 percent fewer errors. How can that kind of design improvement be achieved in general practice?

Design Practice

It seems fair to characterize current user interface software design as art rather than science, depending more upon individual judgment than systematic application of knowledge (Ramsey and Atwood, 1979; 1980). As an art, user interface design is best practiced by experts, by specialists experienced in the human engineering of computer systems. But such experts are not always available to help guide system development, and it is clear that they cannot personally guide every step of design. What is needed is some way to embody expert judgment in the form of explicit design guidelines.

For military information systems, Military Specification MIL-H-48655B (1979) calls for a system development sequence starting with requirements analysis, functional specification and verification before any software design begins. The actual course of user interface software development will sometimes depart from that desired sequence. There may be no explicit attempt to determine user interface requirements. Specifications may include only rudimentary references to user interface design, with general statements that the system must be "easy to use." In the absence of effective guidance, both the design and implementation of user interface software may become the responsibility of programmers unfamiliar with operational requirements. Detection and correction of design flaws may occur only after system prototyping, when software changes are difficult to make.

Human engineering standards and design handbooks have in the past been of little use to the software designer. The popular human factors design handbook by Woodson (1981) is typical. Its nearly 1000 pages include only three pages of general material on information processing, and there is no reference to computer systems in its index.

MIL-STD-1472B (1974), for many years the major human engineering design standard for military system procurement, was concerned almost exclusively with hardware design and physical safety. In 1981, MIL-STD-1472 was published in a revised "C" version. That version included nine pages dealing with user interface software design, in a section titled "Personnel-Computer Interface." That material was later expanded to 19 pages, titled "User-Computer Interface," in a revision of MIL-STD-1472C (1983). Thus a beginning has been made, but much more is needed. The question is, what guidance can be offered for user interface software design?

Design Guidelines

Until several years ago, there had been no serious attempt to integrate the scattered papers, articles and technical reports that constitute the literature of user-computer interaction. A first step was made, under sponsorship of the Office of Naval Research (ONR), in compilation of an extensive bibliography on this subject (Ramsey, Atwood and Kirshbaum, 1978). A significant follow-on effort culminated in publication by Ramsey and Atwood (1979) of a comprehensive summary of this literature.

In reviewing the literature, it is apparent that most published reports dealing with the user-computer interface describe applications rather than design principles. A popular early book on the design of user-computer dialogues offered stimulating examples, covering a range of on-line applications, but was disappointing in its failure to emphasize design principles (Martin, 1973). The ONR bibliography cited above includes 564 items, but identifies only 17 as offering design guidelines.

Although accepted principles for user interface design have not been available, some work has been accomplished toward that end. As experience has been gained in the use of on-line computer systems, some experts have attempted to set forth principles ("guidelines," "ground rules," "rules of thumb") for design of the user-computer interface. If experts cannot yet assert tested principles for user interface design, they might still offer sensible recommendations as a guide for designers.

Military agencies are not the only organizations seeking guidelines for user interface design. There is keen interest in this topic within industrial and commercial organizations, and throughout the general community of people who develop and use information systems. David Penniman, writing for the User On-Line Interaction Group of the American Society for Information Sciences, has cited the need for "an interim set of guidelines for user interface design based on available literature and pending the development of better guidelines as our knowledge increases" (1979, page 2). Penniman goes on to remind us that interim guidelines are better than no guidelines at all.

In a survey of people concerned with user interface design (Smith and Mosier, 1984b), respondents generally support Penniman's activist position. Given a choice between trying to develop a complete set of user interface guidelines now, when many of them must be based on judgment rather than experimental data, or else accepting only a partial set of guidelines based on evaluated research, most respondents would go with judgment now.

It is clear, of course, that system developers cannot wait for future research data in making present design decisions. To meet current needs, several in-house handbooks have been published to guide user interface design within particular organizations (NASA, 1979; Galitz, 1980; Brown, Brown, Burkleo, Mangelsdorf, Olsen, and Perkins, 1983; Sidorsky, Parrish, Gates, and Munger, 1984). These in-house guidelines draw heavily from those in earlier publications, especially the influential IBM report by Engel and Granda (1975), as modified by the authors' own good judgment.

The ESD/MITRE compilation of user interface design guidelines over the past several years has drawn from the work of our predecessors, and will help support the work of others to follow. Each year our guidelines compilation has grown larger. In this present report there are 944 guidelines. This compilation represents the most comprehensive guidance available for designing user interface software, and for that reason this report is recommended as a basic reference for developing information systems.

Guidelines Organization

In the numbered sections of this report, guidelines are organized within six functional areas of user-system interaction:

Section	Functional Area	Number of Guidelines
1	Data Entry	199
2	Data Display	298
3	Sequence Control	184
4	User Guidance	110
5	Data Transmission	83
6	Data Protection	70

Each section of guidelines covers a different functional area of user-system interaction, although there is necessarily some overlap in topical coverage from one section to another. Within each section, guidelines are grouped by specific functions. Each function has its own numeric designator, as listed in the table of contents for this report.

In adopting this functional organization, we have established a broad conceptual structure for dealing with the range of topics that must be considered in user interface design. Such a conceptual structure is urgently needed to help clarify discourse in this field.

Each section of the guidelines begins with an introductory discussion of design issues relating to the general functional area. That discussion provides some perspective for the guidelines that follow. The discussion concludes with brief definitions of the various user interface functions covered in that section of the guidelines, along with an internal table of contents for that section, which may help to lead a reader directly to functions of immediate interest.

Function definitions are repeated in boxed format to begin the listing of guidelines under each function. Those definitions should aid reader understanding of the material, and the boxed format will provide a notable visual indicator that a new series of guidelines has begun.

The guidelines themselves are numbered sequentially under each function, in order to permit convenient referencing. Under any function there will usually be guidelines pertaining to various subordinate topics. Each guideline has been given a short title to indicate its particular subject matter. Sometimes one guideline may introduce a new topic and then be followed by several closely related guidelines. Each of those related guidelines has been marked with an arrow (▶) next to its title.

Following its number and title, each guideline is stated as a single sentence. Guidelines are worded as simply as possible, usually in general terms to permit broad application, but sometimes with contingent phrasing intended to define a more limited scope of application.

In many instances, a stated guideline will be illustrated by one or more examples. When an example includes some sort of imagined computer output, such as an error message, prompt, menu, etc., that output is printed here in a different typeface: `sample computer output`.

There is no question that specific examples can help clarify a generally-worded guideline. Sometimes a reader will say, "I didn't really understand the guideline until I saw the example." But there is a potential hazard in examples. Because any example must be narrowly specific, a reader who relies on that example may interpret the guideline as having a narrower meaning than was intended. It is important to emphasize that examples are presented here only to illustrate the guidelines, and are **not** intended to limit the interpretation of guidelines.

Where the validity of a guideline is contingent upon special circumstances, examples may be followed by noted exceptions. Those exceptions **are** intended to limit the interpretation of a guideline.

Where further clarification of a guideline seems needed, examples and noted exceptions are followed by supplementary comments. Those comments may explain the reasoning behind a guideline, or suggest possible ways to interpret a guideline, or perhaps note relations between one guideline and another.

Where a guideline has been derived from or is related in some way to other published reports, a reference note may be added citing author(s) and date. Complete citations for those references are listed following Section 6 of the guidelines. Where a guideline corresponds with other published design standards or guidelines, which is often the case, reference citations are given by letter codes. Those codes are explained in the reference list.

Where a guideline is specifically related to guidelines in other sections, appropriate cross references are given. Those cross references permit an interested reader to explore how a particular topic is dealt with in different sections of the guidelines.

Toward the back of this report, following the guidelines is the reference list. Following the reference list is a glossary. The glossary defines word usage in the guidelines, for those words that are used here differently or more narrowly than in the general literature on user interface design. There is no question that we need more consistent terminology in this field.

Following the glossary is a list of the titles for all 944 guidelines, which may help a reader who is trying to find guidelines that pertain to a particular topic.

Following the list of guideline titles, and concluding this report, is a topical index of the guidelines material. That index is intended to help readers find guidelines on a particular subject, independently of the functional organization that has been imposed on the guidelines material.

These notes on organization and format should serve to allow a student of the subject to skim the guidelines material and find information on different topics. For those readers who seek to apply the guidelines in software design, some further comments are needed.

Applying the Guidelines

Not all of the guidelines proposed here can be applied in designing any particular system. For any particular system application, some of the guidelines will be relevant and some will not. In a recent survey of guidelines application (Mosier and Smith, 1986), respondents indicated that they actually applied only 40 percent of the guidelines published in a previous report.

There is another problem to consider. Design guidelines such as those proposed here must be generally worded so that they might apply to many different system applications. Thus generally-worded guidelines must be translated into specific design rules before they can actually be applied.

The process of selecting relevant guidelines for application and translating them into specific design rules is referred to here as "tailoring." Who will do this guidelines tailoring? It should be the joint responsibility of system analysts and human factors specialists assessing design requirements, of software designers assessing feasibility, and of their managers. It may also be helpful to include representatives of the intended system users in this process, to ensure that proposed design features will meet operational requirements. To simplify discussion, we shall call all of these persons "designers."

As a first step in guidelines tailoring, a designer must review this report in order to identify those guidelines that are relevant. For example, if an application will require menus, then the 36 guidelines in Section 3.1.3 dealing with Menu Selection are potentially relevant. For a large information system, the list of relevant guidelines may be quite large.

Once all relevant guidelines have been identified, a designer must review them and decide which ones actually to apply. There are two reasons why a designer might not wish to apply all relevant guidelines. First, for any given application some guidelines may conflict, and the designer must therefore choose which are more important. Second, budgetary and time restrictions may force the designer to apply only the most important guidelines — those that promise to have the greatest effect on system usability.

As noted above, because guidelines are intended for use on a variety of systems they are worded in general terms. Before a guideline can actually be applied it must be translated into specific design rules. For instance, a guideline which states that displays should be consistently formatted might be translated into design rules that specify where various display features should appear, such as the display title, prompts and other user guidance, error messages, command entries, etc.

Any guideline can have different possible translations. A guideline which states that each display should be uniquely identified could be translated into a design rule that display titles will be bolded and centered in the top line of the display. Or it could be translated into a design rule that display titles will be capitalized in the upper left corner of the display.

What would happen if guidelines were **not** translated into design rules, but instead were given directly to interface designers? If designers do not decide as a group what design rules will be used, then each designer will decide separately in the course of applying guidelines. The result will surely be an inconsistent design.

After design rules have been specified for each selected guideline, those rules should be documented for reference by software designers and others involved in system development. Documentation of agreed rules, subject to periodic review and revision as necessary, will help coordinate the design process. Documented rules can then be applied consistently for a given application. With appropriate modifications, rules adopted for one application might later be used for other applications.

In the course of design, it may be determined that a particular design rule cannot be used. Therefore, some means must be provided to deal with exceptions. If a design rule is not appropriate for one particular display, then an exception can be made by whoever has been appointed to make such decisions. But if a design rule cannot be implemented at all, perhaps due to other design constraints, then all designers for that particular system must be notified, and perhaps another design rule must be substituted.

Finally, after the design is complete, it must be evaluated against the original design requirements to ensure that all design rules have indeed been followed. To help in the exception process and in the evaluation process, it may be useful to assign different weights to the various rules, indicating which are more important than others. Such weighting will help resolve the trade-offs that are an inevitable part of the design process.

Role of Guidelines in System Development

If guidelines are applied in the way described here, there are some significant implications for the role of guidelines in system development. Generally stated guidelines should be offered to designers as a potential resource, rather than imposed as a contractual design standard (Smith, 1986). It is only specifically worded design rules that can be enforced, not guidelines.

Design rules can be derived from the guidelines material, but that conversion from guidelines to rules should be performed as an integral part of the design process, serving to focus attention on critical design issues and to establish specific design requirements. Once agreed design rules are established, those rules can be maintained and enforced by the managers of system development projects.

Specific design rules probably cannot be imposed effectively at the outset of system development by some external agency — by a sponsoring organization or by a marketing group. It is the **process** of establishing design rules that should be imposed, rather than the rules themselves. A software design contractor might reasonably be required to establish rules for the design of user interface software, subject to review by the contracting agency. Available guidelines could be cited as a potentially useful reference for that purpose.

Some other cautionary comments about the application of guidelines deserve consideration here. Guidelines in themselves cannot assure good design for a variety of reasons (Thimbleby, 1985). Guidelines cannot take the place of experience. An experienced designer, one skilled in the art, might do well without any guidelines. An inexperienced designer might do poorly even with guidelines. Few designers will find time to read an entire book of guidelines. If they do, they will find it difficult to digest and remember all of the material. If guidelines and/or the rules derived from guidelines are to be helpful, they must be kept continually available for ready reference.

Guidelines cannot take the place of expert design consultants, or at least not entirely. A design expert will know more about a specific topic than can be presented in the guidelines. An expert will know what questions to ask, as well as many of the answers. An expert will know how to adapt generally-stated guidelines to the specific needs of a particular system design application. An expert will know how to trade off the competing demands of different guidelines, in terms of operational requirements.

For maximum effectiveness, guideline tailoring must take place early in the design process before any actual design of user interface software. In order to tailor guidelines, designers must have a thorough understanding of task requirements and user characteristics. Thus task analysis is a necessary prerequisite of guidelines tailoring.

The result of guidelines application will be a design for user interface software that may incorporate many good recommendations. However, even the most careful design will require testing with actual users in order to confirm the value of good features and discover what bad features may have been overlooked. Thus prototype testing must follow initial design, followed in turn by possible redesign and operational testing.

Indeed, testing is so essential for ensuring good design that some experts advocate early creation of an operational prototype to evaluate interface design concepts interactively with users, with iterative design changes to discover what works best (Gould and Lewis, 1983). But prototyping is no substitute for careful design. Prototyping will allow rapid change in a proposed interface; however, unless the intial design is reasonably good, prototyping may not produce a usable final design.

Considering the system development process overall, guidelines application will not necessarily save work in user interface design, and in fact may entail extra work, at least in the initial stage of establishing design rules. But guidelines application should help produce a better user interface. Because guidelines are based on what is known about good design, the resulting user interface is more likely to be usable. Certainly the common application of design rules by all designers working on a system should result in more consistent user interface design. And the single objective on which experts agree is design consistency.

Section 1
DATA ENTRY

DATA ENTRY

Data entry refers to user actions involving input of data to a computer, and computer responses to such inputs. The simplest kind of data entry consists merely of pointing at something — selecting an item or designating a position on a computer-generated display. In more complicated modes of data entry, a user may have to control the format of data inputs as well as their contents. Thus questions of format control in text entry/editing and graphic interaction may properly be considered questions of data entry.

Note, however, that user inputs which initiate or interrupt transactions — such as command entries, or control entries selected from a displayed menu or by function keys — pose rather different questions of design. Such control entries are discussed in Section 3 of these guidelines.

Data can be entered into a computer in a variety of different ways. Users might designate position or direction by pointing at a display. Users might enter numbers, letters, or more extended textual material by keyed inputs, or in some applications by spoken inputs. Data might be keyed into displayed forms or tables, into constrained message formats, or as free text. In graphic interaction users might draw pictures or manipulate displayed graphic elements. These different types of data entry all merit consideration here.

The computer will also play a role in the data entry process, guiding users who need help, checking data entries to detect errors, and providing other kinds of data processing aids. A designer of user interface software must be concerned about computer processing logic as well as data input by the user.

Data entry is heavily emphasized in clerical jobs, and many other jobs involve data entry to some degree. Because data entry is so common, and because inefficiencies caused by poorly designed data entry transactions are so apparent, many published recommendations for good user interface design deal with data entry questions. Human factors specialists can probably give better advice about data entry than about any other functional area of user interface design.

Data entry requires hardware, and the proper design of input devices has received considerable attention, including concern for standardization of keyboard layouts. Future advances in hardware design may well influence data entry tasks, as suggested by current advocacy of voice input.

But the major need in today's information systems is for improving the **logic** of data entry, and it is there that design guidance should prove most helpful. Thus the guidelines presented here deal with data entry functions, insofar as possible, without regard to their hardware implementation.

The general objectives of designing data entry functions are to establish consistency of data entry transactions, minimize input actions and memory load on the user, ensure compatibility of data entry with data display, and provide flexibility of user control of data entry. Stated in such general terms, these principles do not provide helpful guidance to designers. Somehow these general ideas must be converted into more specific guidelines.

Introduction

The process of converting general principles into more detailed guidelines will lead to a considerable proliferation of ideas. With regard to minimizing input actions, one guideline might be that a user should not have to enter the same data twice. Probably every designer knows that, even if it is sometimes forgotten. A related guideline might be that a user should not have to enter data already entered by another user. That seems to make good sense, although one could imagine occasional exceptions when cross validation of data inputs is required.

How can duplicative data entry be avoided in practice? The solution lies in designing the user interface (programming the computer) to maintain **context**. Thus when a user identifies a data category of interest, say a squadron of aircraft, the computer should be able to access all previously entered data relevant to that squadron and not require the user to enter such data again.

In repetitive data entry transactions the user should have some means of establishing context. One method is to allow users to define default entries for selected data items, in effect telling the computer that those items will stay the same until the default value is changed or removed. If a user enters one item of data about a particular squadron, it should be possible to enter other items thereafter without having to re-identify that squadron.

Context should also be preserved to speed correction of input errors. One significant advantage of on-line data entry is the opportunity for immediate computer validation of user inputs, with timely feedback so that a user can correct errors while the data are still fresh in mind and while documented source data are still at hand. Here the computer should preserve the context of each data entry transaction, saving correct items so that the user does not have to enter those again while changing incorrect items.

Preservation of context is, of course, important in all aspects of user-system interaction, with implications for data display, sequence control and user guidance, as well as for data entry. The importance of context is emphasized again in the discussion of those other functional areas.

Another important design concept is **flexibility**. It is easy to say that the interface should adapt flexibly to user needs, but the specific means of achieving such flexibility must be spelled out in design guidelines. For data entry functions it is important that the pacing of inputs be controlled flexibly by the user. Tasks where the pacing of user inputs is set by a machine are stressful and error-prone.

Aside from flexibility in pacing, users will often benefit from having some flexible choice in the ordering of inputs. What is needed for interface design is some sort of suspense file to permit flexible ordering of data entries, including temporary omission of unknown items, backup to correct mistaken entries, cancellation of incomplete transactions, etc.

As noted above, users may also benefit from flexibility in defining default options to simplify data entry during a sequence of transactions. Some systems include only those defaults anticipated by the designers, which may not prove helpful to the user in a particular instance. Thus the concept of flexibility is related to maintaining context, and is related also to many other aspects of interface design.

Introduction

The guidelines proposed here deal with data entry in terms of specific functions, covering different kinds of data entry and different kinds of computer processing support. Some topics, such as "abbreviation", which pertain to all data entry are covered in an initial group of guidelines dealing generally with the subject. A summary of the functional coverage in this section is presented on the next page. These guidelines recommend specific ways to accomplish the fundamental design objectives for data entry.

Objectives:

Consistency of data entry transactions

Minimal entry actions by user

Minimal memory load on user

Compatibility of data entry with data display

Flexibility for user control of data entry

Functions

Data entry refers to user actions involving input of data to a computer, and computer responses to such inputs.

Data Entered Only Once •1

Ensure that a user need enter any particular data only once, and that the computer can access those data if needed thereafter for the same task or for different tasks.

COMMENT: In effect, this recommendation urges integrated and flexible software design so that different programs can access previously entered data as needed. Requiring re-entry of data would impose duplicative effort on users and increase the possibility of entry errors.

SEE ALSO: 1.8•9.

Entry via Primary Display •2

When data entry is a significant part of a user's task, entered data should appear on the user's primary display.

EXAMPLE: As a negative example, entry via typewriter is acceptable only if the typewriter itself, under computer control, is the primary display medium.

COMMENT: When the primary display is basically formatted for other purposes, such as a graphic display for process control, a separate window on the display may have to be reserved for data entry.

Feedback During Data Entry •3

Provide displayed feedback for all user actions during data entry; display keyed entries stroke by stroke.

EXCEPTION: For reasons of data protection, it may **not** be desirable to display passwords and other secure entries.

REFERENCE: EG 6.3.7; MS 5.15.2.1.2, 5.15.2.2.3.

SEE ALSO: 3.0•14, 4.2•1.

1.0 General

•4 ▶ **Fast Response**

Ensure that the computer will acknowledge data entry actions rapidly, so that users are not slowed or paced by delays in computer response; for normal operation, delays in displayed feedback should not exceed 0.2 seconds.

EXAMPLE: A key press should be followed by seemingly immediate display of its associated symbol, or by some other appropriate display change.

COMMENT: This recommendation is intended to ensure efficient operation in routine, repetitive data entry tasks. Longer delays may be tolerable in special circumstances, perhaps to reduce variability in computer response, or perhaps in cases where data entry comprises a relatively small portion of the user's task.

COMMENT: Note that this guideline refers to acknowledgment, rather than final processing of entries which may be deferred pending an explicit ENTER action.

REFERENCE: EG Table 2.

SEE ALSO: 3.0•18, 3.0•19.

•5 **Single Method for Entering Data**

Design the data entry transactions and associated displays so that a user can stay with one method of entry, and not have to shift to another.

EXAMPLE: Minimize shifts from lightpen to keyboard entry and then back again.

EXAMPLE: As a negative example, a user should **not** have to shift from one keyboard to another, or move from one work station to another, to accomplish different data entry tasks.

COMMENT: This, like other guidelines here, assumes a task-oriented user, busy or even overloaded, who needs efficiency of data entry.

REFERENCE: BB 2.11; EG 6.1.1; Foley and Wallace, 1974; Shneiderman, 1982.

SEE ALSO: 1.1•14.

Defined Display Areas for Data Entry •6

Where data entry on an electronic display is permitted only in certain areas, as in form filling, provide clear visual definition of the entry fields.

EXAMPLE: Data entry fields might be underlined, or perhaps highlighted by reverse video.

EXCEPTION: For general text entry of variable (unrestricted) length, no field delimiters are needed. In effect, keyed text entries can replace nothing (null characters).

COMMENT: Display formats with field delimiters provide explicit user guidance as to the location and extent of data entry fields. Where delimiters extend throughout an entry field, as in underlining, then any keyed data entries should replace the delimiter characters on the display.

REFERENCE: BB 2.2.1.

SEE ALSO: 1.4•10.

Consistent Method for Data Change •7

In keyed data entry, always allow users to change previous entries if necessary (including displayed default values) by delete and insert actions; if data change is sometimes made by direct character substitution ("typeover"), then that option should also be consistently available.

EXAMPLE: Form filling may require typeover to replace displayed characters such as underscores that act as field delimiters.

COMMENT: Text editing on an electronic display can be handled with or without typeover; there seems to be no published research on the relative efficiency of user performance under these two conditions.

COMMENT: Using typeover, there is some risk of user confusion in replacement of an old value with a new one, during the transitional period when the item being changed is seen as a composite beginning with the new value and ending with the old. Some designers do not permit overtyping for that reason.

COMMENT: In some applications it may help the user to key a new entry directly above or below display of the prior entry it will replace, if that is done consistently. Here the user can compare values before confirming entry of the new data and deletion of the old.

REFERENCE: BB 2.10; Keister and Gallaway, 1983.

•8 **User-Paced Data Entry**

Allow users to pace their data entry, rather than having the pace being controlled by computer processing or external events.

COMMENT: The timing of user-paced data entry will fluctuate depending upon a user's momentary needs, attention span and time available. At maximum speed, user-paced performance is more accurate than that achieved by machine pacing.

COMMENT: When user pacing does not seem feasible, as in some real-time process control applications, reconsider the general approach to task allocation and interface design.

REFERENCE: MS 5.15.2.1.1; Bertelson, Boons and Renkin, 1965.

•9 **Explicit ENTER Action**

Always require a user to take an explicit ENTER action to initiate processing of entered data; do **not** initiate processing as a side effect of some other action.

EXAMPLE: As a negative example, returning to a menu of control options should **not** by itself result in computer processing of data just keyed onto a display.

EXCEPTION: In routine, repetitive data entry transactions, successful completion of one entry may automatically lead to initiation of the next, as in keying ZIP codes at an automated post office.

COMMENT: Deferring processing until after an explicit ENTER action will permit a user to review data and correct errors before computer processing, particularly helpful when data entry is complex and/or difficult to reverse.

REFERENCE: MS 5.15.2.1.4.

SEE ALSO: 1.4•1, 1.4•2, 3.0•5, 4.0•2, 6.0•9, 6.3•5.

•10 ▶ **ENTER Key Labeling**

Label an ENTER key explicitly to indicate its function.

EXAMPLE: As a negative example, the ENTER key should **not** be labeled in terms of mechanism, such as CR or RETURN or XMIT.

COMMENT: For a novice computer user, the label should perhaps be even more explicit, such as ENTER DATA. Ideally, one consistent ENTER label would be adopted for all systems and so become familiar to all users.

COMMENT: Some other label might serve as well, if it were used consistently. In some current systems the ENTER key is labeled GO or DO, implying a generalized command to the computer, "Go off and do it."

REFERENCE: PR 3.3.9.

SEE ALSO: 3.0•16, 4.0•10.

Explicit CANCEL Action •11

Require a user to take an explicit action in order to cancel a data entry; data cancellation should **not** be accomplished as a side effect of some other action.

EXAMPLE: As a negative example, casual interruptions of a data entry sequence, such as paging through forms, or detouring to HELP displays, should **not** have the effect of erasing partially completed data entries.

COMMENT: If a requested sequence control action implies a more definite interruption, such as a LOG-OFF command, or a command to return to a menu display, then the user should be asked to confirm that action and alerted to the loss of any data entries that would result.

SEE ALSO: Section 3.3.

Feedback for Completion of Data Entry •12

Ensure that the computer will acknowledge completion of a data entry transaction with a confirmation message, if data entry was successful, or else with an error message.

EXCEPTION: In a sequence of routine, repetitive data entry transactions, successful completion of one entry might result simply in regeneration of the initial (empty) data entry display, in order to speed the next entry in the sequence.

COMMENT: Successful data entry should **not** be signaled merely by automatic erasure of entered data from the display, except possibly in the case of repetitive data entries. For single data entry transactions, it may be better to leave entered data on the display until the user takes an explicit action to clear the display.

REFERENCE: MS 5.15.5.4.

SEE ALSO: 1.0•3, 3.0•14, 4.2•1.

•13 ▶ **Feedback for Repetitive Data Entries**

For a repetitive data entry task that is accomplished as a continuing series of transactions, indicate successful entry by regenerating the data entry display, automatically removing the just-entered data in preparation for the next entry.

COMMENT: Automatic erasure of entered data represents an exception to the general principle of control by explicit user action. The interface designer may adopt this approach, in the interests of efficiency, for data entry transactions that task analysis indicates will be performed repetitively.

COMMENT: In addition to erasure of entered data, a message confirming successful data entry might be displayed. Such a message may reassure uncertain users, especially in system applications where computer performance is unreliable.

REFERENCE: EG 4.2.10.

SEE ALSO: 1.0•3, 3.0•14, 4.2•1.

•14 ▶ **Feedback when Changing Data**

If a user requests change (or deletion) of a data item that is not currently being displayed, offer the user the option of displaying the old value before confirming the change.

EXCEPTION: Expert users may sometimes wish to implement data changes without displayed feedback, as in global replace transactions, accepting the attendant risk.

COMMENT: Displayed feedback will help prevent inadvertent data change, and is particularly useful in protecting delete actions. Like other recommendations intended to reduce error, it assumes that accuracy of data entry is worth extra effort by the user. For some tasks, that may not be true.

SEE ALSO: 6.3•16.

•15 **Keeping Data Items Short**

For coded data, numbers, etc., keep data entries short, so that the length of an individual item will not exceed 5-7 characters.

EXAMPLE: Coded data may include such items as badge numbers, payroll numbers, mail stops, equipment and part numbers, etc.

COMMENT: For coded data, lengthy items may exceed a user's memory span, inducing errors in both data entry and data review.The nine-digit ZIP codes proposed by the US Postal Service will prove difficult to remember accurately.

COMMENT: Proper names, meaningful words, and other textual material are not coded data. Such items can be remembered more easily, and the length restriction recommended here need not apply.

REFERENCE: BB 1.5.2; EG 6.3.3.

▶ Partitioning Long Data Items •16

When a long data item must be entered, it should be partitioned into shorter symbol groups for both entry and display.

EXAMPLE: A 10-digit telephone number can be entered as three groups, NNN-NNN-NNNN.

REFERENCE: BB 1.4.1; MS 5.15.3.1.7, 5.15.3.5.7, 5.15.3.5.8; Wright, 1977.

SEE ALSO: 2.2•14.

Optional Abbreviation •17

Allow optional abbreviation of lengthy data items to minimize data entry keying by expert users, when that can be done without ambiguity.

COMMENT: Novice and/or occasional users may prefer to make full-form entries, while experienced users will learn and benefit from appropriate abbreviations.

REFERENCE: BB 2.4.1; EG 6.3.5; MS 5.15.2.2.7.

▶ Distinctive Abbreviation •18

When defining abbreviations or other codes to shorten data entry, choose them to be distinctive in order to avoid confusing similarity with one another.

EXAMPLE: BOS vs. LAS is good; but LAX vs. LAS risks confusion.

REFERENCE: BB 3.1; MS 5.15.2.1.10.

▶ Simple Abbreviation Rule •19

When defining abbreviations, follow some simple abbreviation rule and ensure that users understand that rule.

EXAMPLE: Simple truncation is probably the best rule when that can be done without ambiguity.

COMMENT: When encoding abbreviations for data entry the user must know what the rule is. Truncation provides novice users with a straightforward and highly successful method for generating abbreviations, and is a rule that can be easily explained. Moreover, truncation works at least as well, and often better than, more complicated rules, such as word contraction with omission of vowels.

COMMENT: Designers of military systems may wish to consult the relevant standard for abbreviations, MIL-STD-12D.

REFERENCE: Ehrenreich, 1985; Ehrenreich and Porcu, 1982; Hirsh-Pasek, Nudelman and Schneider, 1982; Moses and Ehrenreich, 1981.

1.0 General

•20

▶ Minimal Exceptions to Abbreviation Rule

Use special abbreviations (i.e., those not formed by consistent rule) only when they are required for clarity.

COMMENT: Special abbreviations will sometimes be needed to distinguish between words whose abbreviations by rule are identical, or when abbreviation by rule forms another word, or when the special abbreviation is already familiar to system users.If more than 10 percent of abbreviations are special cases, consider changing the abbreviation rule.

REFERENCE: Moses and Ehrenreich, 1981.

•21

▶ Minimal Deviation from Abbreviation Rule

When an abbreviation must deviate from the consistent rule, minimize the extent of deviation.

EXAMPLE: In abbreviation by truncation, letters in the truncated form should be changed one at a time until a unique abbreviation is achieved.

REFERENCE: Moses and Ehrenreich, 1981.

•22

▶ Fixed Abbreviation Length

Make abbreviations the same length, the shortest possible that will ensure unique abbreviations.

COMMENT: Desirable length will depend upon the vocabulary size of words to be abbreviated. For a vocabulary of 75 words, 4-letter abbreviations might suffice. For smaller vocabularies, still shorter abbreviations might be used.

REFERENCE: Moses and Ehrenreich, 1981.

•23

▶ Clarifying Unrecognized Abbreviations

When the computer cannot recognize an abbreviated data entry, question the user as necessary to resolve any ambiguity.

EXAMPLE: This may occur when a user enters a misremembered abbreviation.

Prompting Data Entry •24

Provide prompting for the required formats and acceptable values for data entries.

EXAMPLE: (Good)
```
Vehicle type: __
  c = Car
  t = Truck
  b = Bus
```
(Bad)
```
Vehicle type: __
```

EXCEPTION: Prompting may not be needed by skilled users and indeed may hinder rather than help their performanee in situations where display output is slow (as with Teletype displays); for such users prompting might be provided as an **optional** aid.

COMMENT: Prompting is particularly needed for coded data entries. Menu selection may be appropriate for that purpose, because menu selection does not require the user to remember codes but merely to choose among displayed alternatives. Other methods of prompting inelude labeling data fields, sueh as

```
Vehicle type (c/t/b): __
```

and/or providing optional guidance displays.

REFERENCE: Gade, Fields, Maisano, Marshall, and Alderman, 1981; Seibel, 1972.

SEE ALSO: 1.4•5, 4.4•7, and Seetion 3.1.3.

Character Entry via Single Keystroke •25

Allow users to enter each character of a data item with a single stroke of an appropriately labeled key.

EXAMPLE: As a negative example, when a keyboard is intended primarily for numerie input, with several letters grouped on each key sueh as a telephone keypad, do **not** require a user to make alphabetic entries by double keying.

COMMENT: Deviees that involve eomplex keying methods for alphabetie entry (e.g., pressing more than one key, simultaneously or sueeessively) require special user training and risk frequent data entry errors.

COMMENT: When hardware limitations such as those of a telephone keypad seem to require double keying of alphabetic entries, try to limit data eodes so that only single-keyed (numerie) entries are required. Alternatively, eonsider providing software to interrogate the user to resolve any ambiguities resulting from single-keyed alphabetic entries.

REFERENCE: Butterbaugh and Roekwell, 1982; Smith and Goodwin, 1971a.

•26 ▶ **Minimal Shift Keying**

Design data entry transactions to minimize the need for shift keying.

COMMENT: Shift keying can be considered a form of double keying, which imposes a demand for extra user attention. Keyboard designers should put frequently used characters where they can be easily keyed. Conversely, software designers should avoid frequent use of characters requiring shift keying.

REFERENCE: EG 6.3.12.

•27 **Upper and Lower Case Equivalent**

For coded data entry, treat upper and lower case letters as equivalent.

COMMENT: For data codes, users find it difficult to remember whether upper or lower case letters are required, and so the software design should not try to make such a distinction. For text entry, however, conventional use of capitalized letters should be maintained.

SEE ALSO: 1.3•10, 3.0•12.

•28 **Decimal Point Optional**

Allow optional entry or omission of a decimal point at the end of an integer as equivalent alternatives.

EXAMPLE: An entry of "56." should be processed as equivalent to an entry of "56", and vice versa.

COMMENT: If a decimal point is required for data processing, the computer should probably be programmed to append one as needed. Most users will forget to do it.

REFERENCE: Keister and Gallaway, 1983.

•29 **Leading Zeros Optional**

For general numeric data, allow optional entry or omission of leading zeros as equivalent alternatives.

EXAMPLE: If a user enters "56" in a field that is four characters long, the system should recognize that entry rather than requiring an entry of "0056".

EXCEPTION: Special cases may represent exceptions to this rule, such as entry of serial numbers or other numeric identifiers.

REFERENCE: BB 2.2.3; EG 6.3.11.

Single and Multiple Blanks Equivalent •30

Treat single and multiple blank characters as equivalent in data entry; do **not** require users to count blanks.

COMMENT: People cannot be relied upon to pay careful attention to such details. The computer should handle them automatically, e.g., ensuring that two spaces follow every period in text entry (if that is the desired convention), and spacing other data items in accord with whatever format has been defined.

SEE ALSO: 3.1.5•17.

Aids for Entering Hierarchic Data •31

If a user must enter hierarchic data, where some items will be subordinate to others, provide computer aids to help the user specify relations in the hierarchic structure.

COMMENT: For simple data structures, question-and-answer dialogues or form filling may suffice to maintain necessary data relations. For more complex data structures, such as those involved in graphic data entry, special techniques may be needed to help users specify the relations among data entries.

SEE ALSO: 1.6•18, 1.8•12.

Speech Input •32

Consider spoken data input only when data entry cannot be accomplished through more reliable methods such as keyed entry or pointing.

EXAMPLE: Postal workers whose hands are occupied sorting packages might speak ZIP codes into a speech recognition device rather than keying entries.

COMMENT: Current speech recognition devices are not well developed and tend to be error prone. Thus there should be some good reason for choosing speech input over more conventional data entry methods. Speech input might be appropriate if a user cannot use his/her hands, perhaps because of a physical handicap or because the user's hands are needed to accomplish other tasks. Speech input may also be appropriate if a user does not have access to a suitable keyboard, as might be the case if data were being entered by telephone.

•33 ▶ **Limited Vocabulary for Speech Input**

Structure the vocabulary used for spoken data entry so that only a few options are needed for any transaction.

COMMENT: To increase the likelihood that a user's valid entries are correctly identified by the system, the user's vocabulary should be predictable. This does not necessarily mean that the vocabulary must be small, though recognition systems that can only accommodate small vocabularies are more prevalent and less expensive. A vocabulary is predictable when a user's choice of inputs at any given time is small, so that the system will be more likely to make a correct match in interpreting an entry.

•34 ▶ **Phonetically Distinct Vocabulary for Speech Input**

Ensure that the spoken entries needed for any transaction are phonetically distinct from one another.

COMMENT: Words which are easily distinguished by one speech recognition system may be confused by another. Thus system testing should be performed in order to determine what sounds a particular system tends to confuse, and what sounds it can distinguish reliably.

•35 ▶ **Easy Error Correction for Speech Input**

Provide feedback and simple error correction procedures for speech input, so that when a spoken entry has not been correctly recognized by the computer, the user can cancel that entry and speak again.

COMMENT: Simple error correction is particularly important with spoken data entry, since speech recognition systems are prone to error except under carefully controlled conditions.

•36 ▶ **Alternative Entries for Speech Input**

When speech input is the only form of data entry available, allow alternatives for critical entries, so that if the system cannot recognize an entry after repeated attempts another entry can be substituted.

EXAMPLE: "Exit" might be defined as an acceptable substitute for "Finished".

COMMENT: Because speech recognition systems are affected by normal variations in a user's voice, and by changes in the acoustic environment, a spoken entry that was accepted yesterday might not be accepted today. Thus for important entries a user should be able to use an alternative word.

COMMENT: Spelling a word letter-by-letter is not an acceptable alternative, since speech recognition systems may have trouble correctly identifying similar sounding letters.

▸ PAUSE and CONTINUE Options for Speech Input •37

Provide PAUSE and CONTINUE options for speech input, so that a user can stop speaking without having to log off the system.

EXAMPLE: A user may wish to stop speaking data for a time in order to answer a telephone, or to speak with a fellow worker. Users should not have to log off the system every time they wish to say something that is not intended as an entry.

SEE ALSO: 3.3•8, 3.3•9.

1.1 Position Designation

> *Position designation refers to user selection and entry of a position on a display, or of a displayed item.*

•1 **Distinctive Cursor**

For position designation on an electronic display, provide a movable cursor with distinctive visual features (shape, blink, etc.).

EXCEPTION: When position designation involves **only** selection among displayed alternatives, highlighting selected items might be used instead of a separately displayed cursor.

COMMENT: When choosing a cursor shape, consider the general content of the display. For instance, an underscore cursor would be difficult to see on a display of underscored text, or on a graphical display containing many other lines.

COMMENT: If the cursor is changed to denote different functions (e.g., to signal deletion rather than entry), then each different cursor should be distinguishable from the others.

COMMENT: If multiple cursors are used on the same display (e.g., one for alphanumeric entry and one for line drawing), then each cursor should be distinguishable from the others.

REFERENCE: Whitfield, Ball and Bird, 1983.

SEE ALSO: 1.1•17, 4.0•9.

•2 **▸ Nonobscuring Cursor**

Design the cursor so that it does not obscure any other character displayed in the position designated by the cursor.

EXAMPLE: A block cursor might employ brightness inversion ("reverse video") to show any other character that it may be marking.

•3 **▸ Precise Pointing**

When fine accuracy of positioning is required, as in some forms of graphic interaction, design the displayed cursor to include a point designation feature.

EXAMPLE: A cross may suffice (like cross-hairs in a telescope), or perhaps a notched or V-shaped symbol (like a gun sight).

COMMENT: Precise pointing will also require a cursor control device capable of precise manipulation. Touch displays, for example, will **not** permit precise pointing.

REFERENCE: MS 5.15.2.1.8.2; Whitfield, Ball and Bird, 1983.

Explicit Activation •4

Require users to take a separate, explicit action, distinct from cursor positioning, for the actual entry (enabling, activation) of a designated position.

EXCEPTION: For line drawing or tracking tasks the need for rapid, continuous entry may override the need to reduce entry errors.

REFERENCE: MS 5.15.2.5.4; Albert, 1982; Foley and Wallace, 1974; Whitfield, Ball and Bird, 1983.

SEE ALSO: 1.6•4, 3.1.3•6.

Fast Acknowledgement of Entry •5

Ensure that the computer will acknowledge entry of a designated position within 0.2 seconds.

EXAMPLE: Almost any consistently provided display change will suffice to acknowledge pointing actions, such as brightening or flashing a selected character.

COMMENT: In some applications it may be desirable to provide an explicit message indicating that a selection has been made.

REFERENCE: EG Table 2; MS 5.15.8.

SEE ALSO: 1.0•3, 4.2•2, 4.2•10.

Stable Cursor •6

Ensure that the displayed cursor will be stable, i.e., that it will remain where it is placed until moved by the user (or by the computer) to another position.

COMMENT: Some special applications, such as aided tracking, may benefit from computer-controlled cursor movement. The intent of the recommendation here is to avoid unwanted "drift".

REFERENCE: EG 6.1.

1.1 Position Designation

•7 **Responsive Cursor Control**

For arbitrary position designation, moving a cursor from one position to another, design the cursor control to permit both fast movement and accurate placement.

COMMENT: Ideally, when a user moves a pointing device the displayed cursor should appear to move instantly. Rough positioning should take no more than 0.5 seconds for full screen traversal. Fine positioning may require incremental stepping of the cursor, or a control device incorporating a large control/display ratio for small displacements, or a selectable vernier mode of control use. For any given cursor control action, the rate of cursor movement should be constant, i.e., should not change with time.

COMMENT: Slow visual feedback of cursor movement can be particularly irritating when a user is repeatedly pressing a cursor control key, or perhaps holding the key down. In that case, slow feedback will cause the user to misjudge location and move the cursor too far.

•8 **Consistent Incremental Positioning**

When cursor positioning is incremental by discrete steps, design the step size of cursor movement to be consistent horizontally (i.e., in both right and left directions), and consistent vertically (in both up and down directions).

COMMENT: Horizontal and vertical step sizes need not be the same, and in general will **not** be.

•9 **▶ Variable Step Size**

When character size is variable, design the incremental cursor positioning to vary correspondingly, with a step size matching the size of currently selected characters.

•10 **▶ Proportional Spacing**

If proportional spacing is used for displayed text, provide computer logic to make necessary adjustments automatically when the cursor is being positioned for data entry or data change.

EXAMPLE: Automatic proportional spacing is useful for cursor control when editing text composed for typesetting.

EXCEPTION: Manual override may help a user in special cases where automatic spacing is not wanted.

COMMENT: Without automatic computer aids, a user probably will not handle proportional spacing accurately.

Continuous Cursor Positioning •11

For continuous position designation, such as needed for line drawing, provide a continuously operable control (e.g., joystick) rather than requiring a user to take incremental, discrete key actions.

SEE ALSO: Section 1.6.2.

Direct Pointing •12

When position designation is the sole or primary means of data entry, as in selection among displayed alternatives, provide cursor placement by direct pointing (e.g., with a touch display or lightpen) rather than incremental stepping or slewing controls (e.g., keys, joystick, etc.).

REFERENCE: MS 5.15.2.5.1; Albert, 1982; Goodwin, 1975; Shinar, Stern, Bubis and Ingram, 1985.

▶ Large Pointing Area for Option Selection •13

In selection of displayed alternatives, design the acceptable area for pointing (i.e., cursor placement) to be as large as consistently possible, including at least the area of the displayed label plus a half-character distance around the label.

COMMENT: The larger the effective target area, the easier the pointing action will be, and the less risk of error in selecting the wrong label by mistake. Some researchers have recommended a target separation on the display of no less than 6 mm.

REFERENCE: BB 2.12; EG 2.3.13, 6.1.3; Whitfield, Ball and Bird, 1983.

SEE ALSO: 3.1.3•5.

Cursor Control at Keyboard •14

When position designation is required in a task emphasizing keyed data entry, provide cursor control by some device integral to the keyboard (function keys, joystick, "cat", etc.).

COMMENT: Separately manipulated devices (lightpen, "mouse", etc.) will tend to slow the user.

REFERENCE: Foley and Wallace, 1974.

SEE ALSO: 1.0•5.

•15 **Compatible Control of Cursor Movement**

Ensure that control actions for cursor positioning are compatible with movements of the displayed cursor, in terms of control function and labeling.

EXAMPLE: For cursor control by key action, a key labeled with a left-pointing arrow should move the cursor leftward on the display; for cursor control by joystick, leftward movement of the control (or leftward pressure) should result in leftward movement of the cursor; etc.

SEE ALSO: 3.0•16.

•16 **Minimal Use of Multiple Cursors**

Employ multiple cursors on a single display only when they are justified by careful task analysis.

EXAMPLE: Multiple cursors might be useful to mark a user's place when manipulating data in multiple display windows.

EXAMPLE: In graphic interaction, one cursor might be used for line drawing and a different cursor for alphanumeric data entry (labels, etc.).

COMMENT: Multiple cursors may confuse a user, and so require special consideration if advocated in USI design.

•17 **▶ Distinctive Multiple Cursors**

If multiple cursors **are** used, make them visually distinctive from one another.

SEE ALSO: 1.1•1.

•18 **▶ Distinctive Control of Multiple Cursors**

If multiple cursors are controlled by a single device, provide a clear signal to the user to indicate which cursor is currently under control.

•19 **▶ Compatible Control of Multiple Cursors**

If multiple cursors are controlled by different devices, ensure that their separate controls are compatible in operation.

EXAMPLE: Assume that one cursor is moved upward on a display by forward motion of a joystick. Then a second cursor should also be moved upward by forward motion — perhaps by forward motion of a second joystick or by forward motion of a thumbwheel or other device.

REFERENCE: Morrill and Davies, 1961.

SEE ALSO: 3.0•16.

Consistent HOME Position •20

When there is a predefined HOME position for the cursor, which is usually the case, define that position consistently on all displays of a given type.

EXAMPLE: HOME might be in the upper left corner of a text display, or at the first field in a form-filling display, or at the center of a graphic display.

COMMENT: The HOME position of the cursor should also be consistent in the different "windows" or sections of a partitioned display.

REFERENCE: MS 5.15.2.1.8.3.

SEE ALSO: 4.4•16.

Consistent Cursor Placement •21

On the initial appearance of a data entry display, ensure that the cursor will appear automatically at some consistent and useful location.

EXAMPLE: In a form-filling display, the cursor should be placed in the first entry field.

REFERENCE: BB 2.1.4; MS 5.15.4.3.6.

SEE ALSO: 1.4•28, 4.4•16.

Easy Cursor Movement to Data Fields •22

If a cursor must be positioned sequentially in predefined areas, such as displayed data entry fields, ensure that this can be accomplished by simple user action.

EXAMPLE: Programmable tab keys are customarily used for this purpose.

COMMENT: Automatic cursor advance is generally **not** desirable.

REFERENCE: MS 5.15.4.3.6.

SEE ALSO: 1.4•26.

•23 **Display Format Protection**

When there are areas of a display in which data entries cannot be made (blank spaces, protected field labels, etc.), make those areas insensitive to pointing actions, i.e., prevent the cursor from entering those areas.

EXCEPTION: When a user may have to modify display formats, then this automatic format protection can be provided as a general default option subject to user override.

COMMENT: Automatic format protection will generally make cursor positioning easier for a user, since the cursor will not have to be stepped through blank areas, and much routine cursor control can be accomplished with only casual reference to the display.

REFERENCE: BB 1.8.13; EG 7.5; MS 5.15.4.3.12; PR 3.3.2.

SEE ALSO: 1.4•7, 2.0•10, 6.2•5.

•24 **Data Entry Independent of Cursor Placement**

Ensure that an ENTER action for multiple data items results in entry of all items, regardless of where the cursor is placed on the display.

COMMENT: A user may choose to move the cursor back to correct earlier data items, and may not move the cursor forward again. The computer should ignore cursor placement in such cases.

SEE ALSO: 6.3•7.

Direction designation refers to user entry of directional data (azimuth, bearing, heading, etc.) on a display.

Analog Entry of Estimated Direction •1

When direction designation is based on graphic representation, provide some analog means of entry, such as vector rotation on the display and/or a suitably designed rotary switch.

EXAMPLE: A rotary switch might be used to indicate heading estimations for displayed radar trails.

EXCEPTION: When approximate direction designation will suffice, for just eight cardinal points, keyed entry can be used.

COMMENT: For matching the directional elements in a graphic display, an entry device providing a visual analog will prove both faster and more accurate.

REFERENCE: Smith, 1962a.

Keyed Entry of Quantified Direction •2

When designation of direction is based on already quantified data, allow keyed entry.

EXAMPLE: A heading entry might be made from a verbal report in which the direction has already been expressed numerically.

1.3 Text

Text entry refers to the initial entry and subsequent editing of textual material, including messages.

•1 **Adequate Display Capacity**

Ensure that display capacity, i.e., number of lines and line length, is adequate to support efficient performance of text entry/editing tasks.

EXAMPLE: For text editing where the page format of subsequent printed output is critical, the user's terminal should be able to display full pages of text in final output form, which might require a display capacity of 50-60 lines or more.

EXAMPLE: For general text editing where a user might need to make large changes in text, i.e., sometimes moving paragraphs and sections, a display capacity of at least 20 lines should be provided.

EXAMPLE: Where text editing will be limited to local changes, i.e., correcting typos and minor rewording, as few as seven lines of text might be displayed.

COMMENT: A single line of displayed text should **not** be used for text editing. During text editing, a user will need to see some displayed context in order to locate and change various text entries. Displaying only a small portion of text will make a user spend more time moving forward and back in a displayed document to see other parts, will increase load on the user's memory, and will cause users to make more errors.

REFERENCE: Elkerton, Williges, Pittman and Roach, 1982; Neal and Darnell, 1984.

SEE ALSO: 1.3•27.

Editing Capabilities During Text Entry •2

Allow users to do at least some simple editing during text entry without having to invoke a separate edit mode.

EXAMPLE: While entering text, users will need at least some capability for text selection (by cursor movement) and deletion.

COMMENT: The intent of this guideline is not to endorse modeless over moded text editors. In fact, when experienced users perform editing tasks, a moded editor may offer some advantages. However if a moded editor is provided, users should be able to do some simple editing such as correcting typographical errors and making simple word changes without having to invoke that editor.

COMMENT: When users will compose text on-line, consider providing a modeless editor rather than a moded editor. Modeless editors offer some advantages for text composition, when users will frequently alternate between text entry and editing.

REFERENCE: Poller and Garter, 1984.

SEE ALSO: 2.0•9.

Free Cursor Movement •3

For text editing, allow users to move the cursor freely over a displayed page of text to specify items for change, and to make changes directly to the text.

COMMENT: Free cursor movement and changes made directly to the text are characteristics usually associated with so-called screen-based editors and not associated with line- or command-based editors. Screen-based editors are preferred by users and are potentially more efficient.

REFERENCE: MS 5.15.3.8.2; Gould, 1981; Roberts and Moran, 1983; Shneiderman, 1982.

SEE ALSO: 2.7.2•8.

•4 ▶ **Control Entries Distinct from Text**

If control entries are made by keying onto the display, such as by keyed menu selections or commands, ensure that they will be distinguishable from displayed text.

EXAMPLE: Keyed control entries might be made only in a reserved window in the display.

COMMENT: The intent here is to help ensure that a user will not inadvertently enter controls as text, or vice versa. If a command entry is keyed into the body of a text display, perhaps at the end of the last sentence, then a user cannot be certain whether the computer will interpret the command as a text entry or as a control entry.

COMMENT: In applications where the screen cannot display all possible format features (e.g., special fonts), format codes representing those features are usually displayed within the text. It is not practical in such cases to display format codes in a separate window, since a displayed code must mark the text that will be affected by the code. These codes should therefore be highlighted in some way to distinguish them from text.

COMMENT: One way of avoiding the problem altogether is to use function keys rather than command entry to control text editing. To provide a general range of text editing functions, however, many keys will be needed. A practical design approach might be to adopt double-keying logic for all keys on a standard (QWERTY) keyboard, where control-F means FILE a document, control-G means GET a document, etc., and providing appropriate extra labels for those keys.

SEE ALSO: 1.3•26.

•5 **Natural Units of Text**

Allow users to specify segments of text in whatever units are natural for entry/editing.

EXAMPLE: For unformatted ("free") text, natural units will be characters, words, phrases, sentences, paragraphs, and pages; for specially formatted text, such as computer program listings, allow specification of other logical units, including lines, subsections, sections, etc.

▶ Control Entry Based on Units of Text •6

Allow users to specify units of text as modifiers for control entries.

EXAMPLE: Consider two alternative control sequences to delete a four-character word:

(Good) DELETE WORD

(Bad) DELETE DELETE DELETE DELETE

COMMENT: Control entries, whether accomplished by function key, menu selection, or command entry, will be easier and more powerful when a user can specify text in natural units, rather than having to repeat an entry for each text character.

COMMENT: When units of text are modifiers for all control entries, the syntax for those control entries will be easier to learn. Whether a control action is to MOVE or to DELETE, the modifiers to specify text are the same.

REFERENCE: MS 5.15.3.8.4.1, 5.15.3.8.4.2.

SEE ALSO: 3.0•6, 4.0•1.

▶ Highlighting Specified Text •7

When text has been specified to become the subject of control entries, highlight that segment of text in some way to indicate its boundaries.

COMMENT: Text may be specified for various purposes — for underlining or bolding, moving, copying, or deleting. Highlighting provides the user with direct feedback on the extent and content of specified text, reducing the likelihood of specification errors.

SEE ALSO: 4.2•10.

▶ Cursor Movement by Units of Text •8

Allow users to move the cursor by specific units of text, as well as one character at a time.

COMMENT: The time necessary to position a cursor is directly related to the number of control actions required. Incremental cursor movement by character will therefore be inefficient when moving the cursor over large units of text.

COMMENT: Cursor positioning will be easier if appropriate function keys can be provided. A SENTENCE key that allows a user to move directly to the next displayed sentence will be more convenient than some double-keying logic such as CONTROL-S.

SEE ALSO: 1.1•7.

1.3 Text

•9

String Search

Allow users to specify a string of text and request the computer to advance (or back up) the cursor automatically to the next (or last previous) occurrence of that string.

COMMENT: Novice users may prefer to move through a displayed document by units of text, such as by word or paragraph. More experienced users, however, may sometimes wish to specify cursor placement directly. An automatic string search capability will generally speed cursor placement in comparison with incremental positioning, particularly when moving over large portions of a document.

COMMENT: Expert users may wish to incorporate special characters in string search, including format control characters such as those for tabbing, bolding, etc.

REFERENCE: Elkerton, Williges, Pittman, and Roach, 1982.

•10

▶ Upper and Lower Case Equivalent in Search

Unless otherwise specified by a user, treat upper and lower case letters as equivalent in searching text.

EXAMPLE: "STRING", "String", and "string" should all be recognized/accepted by the computer when searching for that word.

COMMENT: In searching for words, users will generally be indifferent to any distinction between upper and lower case. The computer should not compel a distinction that users do not care about and may find difficult to make. In situations when case actually is important, allow users to specify case as a selectable option in string search.

COMMENT: It may also be useful for the computer to ignore such other features as bolding, underlining, parentheses and quotes when searching text.

SEE ALSO: 1.0•27, 3.0•12.

•11

▶ Specifying Case in Search

When case is important, allow users to specify case as a selectable option in string search.

EXAMPLE: When searching a document in which all the headings are capitalized, a user might wish to find a string only when it appears in a heading.

COMMENT: Users may also wish to specify features such as bolding, underlining, and quotes when searching text.

▶ Global Search and Replace •12

When systematic editing changes will be made throughout a long document, consider providing a "global search and replace" capability in which the computer will replace **all** occurrences of one text string with another without requiring the user to confirm each change.

COMMENT: Global search and replace could be designed in two different ways. One user might want the computer to make all changes automatically. Another user might want to review and confirm each change. Ideally, both options should be available.

▶ Case in Global Search and Replace •13

If a global search and replace capability is provided, ensure that each time a string is replaced the case of the new string matches the case of the old string, unless otherwise specified by the user.

EXAMPLE: If a word is replacing the first word in a sentence, the first letter of the new word should be capitalized; if it is replacing a word that is entirely in lower case, then the new word should also be in lower case.

COMMENT: On occasion, however, a user might wish to replace an erroneous lower-case word ("Mitre") with a correctly capitalized version ("MITRE").

Automatic Pagination Aids •14

Provide automatic pagination for text entry/editing, allowing users to specify the page size.

EXCEPTION: For short documents, automatic pagination may not be needed.

▶ User Control of Pagination •15

When automatic pagination is provided, allow users to override that pagination in order to specify page numbers at any point in a document.

EXAMPLE: A user might wish to number the first page of a document "23", or perhaps skip a page number in the middle of a document.

COMMENT: When producing a large document, a user may wish to split it into several separate text files for convenience in editing, and hence need to control the page numbering of those component sections. In general, a user will want flexibility in assembling different computer files to create a composite document.

•16 ▶ **Controlling Integrity of Text Units**

When automatic pagination is provided, allow users to specify the number of lines in a paragraph that will be allowed to stand alone at the top or bottom of a page (i.e., the size of "widows" and "orphans"), and to specify any text that should not be divided between two pages, such as inserted lists or tables.

•17 **Automatic Line Break**

For entry/editing of unformatted text, provide an automatic line break ("carriage return") when text reaches the right margin, with provision for user override.

COMMENT: For specially formatted text, such as computer program listings, users may need to control line structure themselves and hence need to override any automatic line break.Even when entering unformatted text, a user will sometimes wish to specify a new line at some particular point, if only for esthetic reasons.

•18 ▶ **Consistent Word Spacing**

Unless otherwise specified by the user, ensure that entered text is left-justified to maintain constant spacing between words, leaving right margins ragged if that is the result.

SEE ALSO: 2.1•8.

•19 **Hyphenation by Users**

In the entry/editing of text, ensure that automatic pagination and line breaks by the computer keep words intact, and introduce hyphenation only where specified by users.

COMMENT: Where compound words have been hyphenated by a user, the computer might break the compound after a hyphen, for pagination or line breaks, unless otherwise specified by the user.Compound words formed with slashes (e.g., "entry/editing") might be treated in a similar manner.

SEE ALSO: 2.1•9.

Format Control by User •20

Provide easy means for users to specify required format control features during text entry/editing, e.g., to specify margin and tab settings.

EXAMPLE: One convenient method of margin and tab control is to allow users to mark settings on a displayed "ruler" that extends the width of a page and is continuously displayed at the top of the screen.

COMMENT: Required format features will vary depending on the application. For instance, font size may be quite important when composing text for typsetting but unnecessary when editing computer programs. The intent of this guideline is that all required format features should be easy to control, and should take priority in interface design. Any format features which are provided but are optional for the user's task should not be made easy to use at the expense of required format features.

Establishing Predefined Formats •21

When text formats must follow predefined standards, provide the standard format automatically; do **not** rely on users to remember and specify proper formats.

EXAMPLE: Standard formats might be required for letters, memos, or other transmitted messages.

SEE ALSO: 5.1•6.

▶ Storing User-Defined Formats •22

When text formats cannot be predicted in advance, allow users to specify and store for future use the formats that might be needed for particular applications.

EXAMPLE: A special format might be adopted for generating a particular report at periodic intervals.

Moving Text •23

Allow users to select and move text segments from one place to another within a document.

COMMENT: A user should **not** have to re-enter (i.e., rekey) text that is already available to the computer.

COMMENT: One convenient method of allowing the user to both move and copy text is to provide a "cut and paste" facility in which the "cut" text remains in a storage buffer and can be "pasted" more than once. For copying, the user can cut text, paste it back into its original location, and paste it again at a new location.

SEE ALSO: 1.0•1.

1.3 Text

•24 **Storing Frequently Used Text**

Allow users to label and store frequently used text segments, and later to recall (copy into current text) stored segments identified by their assigned labels.

EXAMPLE: Much text processing involves repetitive elements specific to different applications, such as signature blocks, technical terms, long names, formulas or equations.

•25 **Necessary Data Displayed**

Ensure that whatever information a user needs for text entry/editing is available for display, as an annotation to displayed text.

EXAMPLE: A user might wish to see format control characters, such as tab and margin settings.

COMMENT: Required annotation will vary with the application. Some annotation may be so commonly needed that it should be continuously displayed — e.g., document name, page number, indication of control mode (if any), etc. Other annotation might be displayed only at user request — such as document status (date last changed, last printed, etc.) which might be displayed in an optional window overlay, and format control characters which might be visible in an optional display mode.

•26 ▶ **Text Distinct from Annotation**

Ensure that annotations to displayed text are distinguishable from the text itself.

EXAMPLE: Continuous annotation might be displayed in the top and/or bottom lines of a page, separated from the text by blank lines; optional annotation might be displayed in window overlays.

COMMENT: This recommendation refers to text annotations added by users, such as marginal notes on printed displays. Other annotation such as format control characters might be shown in a special display mode where text has been expanded to permit annotation between lines.

SEE ALSO: 1.3•4.

Text Displayed as Printed •27

Allow users to display text exactly as it will be printed.

COMMENT: Accurate display is particularly necessary when the format of printed output is important, as when printing letters, tables, etc.

COMMENT: Ideally, text displays should be able to represent all the features that are provided in printed output, including upper and lower case, underlining, bolding, subscripting, superscripting, special symbols, and different styles and sizes of type. When those features are important, the necessary display capability should be provided.

COMMENT: For special formatting features that are not frequently used, it may be sufficient to use extra symbols to note text features that cannot be directly displayed. In that case, care should be taken that such annotation does not disturb the spacing of displayed text. This may require two display modes, one to show text spacing as it will be printed and the other to show annotations to the text.

COMMENT: A corollary to this recommendation is that changes made to displayed text should appear as a user makes them. Some line-based editors show changes only after a document has been filed and later recalled for display, which does not represent good user interface design.

REFERENCE: Foley and Van Dam, 1982; Gould, 1981.

SEE ALSO: 1.3•1.

Flexible Printing Options •28

In printing text, allow users to select among available output formats (line spacing, margin size, etc.) and to specify the parts of a document to be printed; do **not** require that an entire document be printed.

EXAMPLE: Permit a user to print just those portions of a document that have been changed, perhaps specifying just the first page, or page 17, or the last five pages, etc.

COMMENT: This is particularly important when long documents will be edited. A user should not be required to print an entire 50-page document just because of a change to one page.

•29 **Information on Printing Status**

Inform users concerning the status of requests for printouts.

EXAMPLE: The computer should acknowledge print requests immediately, and might provide a subsequent message to indicate when a printout has been completed if the printer is remote (unobservable) from the user's work station.

EXAMPLE: If there is a queue of documents waiting for printout, a user should be able to get an estimate as to when a particular document will be printed.

COMMENT: If a user is responsible for operating a local printer, the computer might display messages to alert the user of potential malfunctions, e.g., if its paper supply is exhausted, if the paper is not correctly loaded, etc.

SEE ALSO: 3.0•14, 4.2•5.

•30 **Auditory Signals for Alerting Users**

During text entry/editing, provide an auditory signal whenever it is necessary to draw a user's attention to the display.

COMMENT: A touch typist entering text from written copy will often not be looking at the display screen, and therefore may not notice visual indicators of errors or mode changes unless they are accompanied by auditory signals.

COMMENT: Note that in a group environment an auditory signal may distract other workers, and may embarrass the user whose error has been thus advertised. In such a work setting, consider allowing users to disable the auditory signal.

SEE ALSO: 2.6•39.

•31 **Protecting Text During Page Overruns**

When a user is inserting text into a document that has already been paginated, ensure that no text is lost if the user inserts more text than a page can hold.

COMMENT: It is difficult for a user to keep track of page size, particularly if the size of the display screen is less than the full page specified for printed text, which is often the case. A user will often not know when more text has been inserted into a page than there is room for. The computer should accommodate text insertions with automatic repagination.

Confirming Actions in DELETE Mode •32

If a DELETE mode is used, highlight any text specified by a user for deletion and require the user to confirm the DELETE action before the computer will process it.

COMMENT: Requiring a user to confirm actions in DELETE mode is particularly important when the control entries for cursor positioning (e.g., WORD, SENTENCE, PARAGRAPH, PAGE) are also used to specify text for deletion, which is often the case. Users will associate the specification of text units primarily with cursor positioning, which is the most frequent action in text editing. In a DELETE mode, after specifying text units for deletion, a user may press a PARAGRAPH key intending to move to the next paragraph but accidentally delete the next paragraph. Confirmation of DELETE actions will tend to prevent such errors.

COMMENT: An alternative approach to this problem is not to provide a continuing DELETE mode, but instead require double keying to accomplish deletions. In a DELETE mode, a user might press a DELETE key followed by unlimited repetitions of a WORD key (or keys specifying other units of text). With double keying, the user would have to press DELETE before **each** selection of a text unit to be deleted.

SEE ALSO: 1.3•6, 6.0•18, 6.3•19.

1.3 Text

•33 Reversible Actions

Design text editing logic so that any user action is immediately reversible.

EXAMPLE: If a user centers a heading and then decides it would look better flush against the left margin, an UNDO action should reverse the centering and move the heading back to its original location.

EXAMPLE: If a user underlines a paragraph of text and then decides it should be in all capital letters instead, an UNDO action should reverse the underlining. The user should **not** be required to delete the paragraph and retype it just to erase the underscoring.

COMMENT: Reversible actions are particularly important in a text editing environment because text formatting often involves experimentation with features such as underscoring, bolding, and spacing. If users know that they can reverse whatever they do, they will feel more free to delete text and experiment with formatting features.

COMMENT: An UNDO capability is currently available in some interface designs. In some applications, however, this capability is provided through the use of an UNDO key which can only reverse the most recent control action. For text editing, users must be able to reverse such formatting features as centering and bolding at any time. Therefore, if control actions are to be made reversible, an UNDO action should be able to reverse more than the most recent command, perhaps by requiring the user to specify which command to undo, and/or to place the cursor at the location of the format feature that is to be reversed.

COMMENT: When text segments have been deleted, it should be possible to retrieve more than the most recent deletion. Some systems do this by storing all deletions in a special file. Deleted text which the user wishes to retrieve can then be moved from the deletion file to the file in which the user is presently working.

REFERENCE: Lee and Lochovsky, 1983; Nicherson and Pew, 1971; Shneiderman, 1982.

SEE ALSO: 3.5•10, 6.0•21.

User Confirmation of Editing Changes •34

When a user signals completion of document editing, allow the user to confirm that changes should be made to the original document, or else to choose alternative options.

COMMENT: A user will generally wish to replace the original document with its edited version. However, sometimes a user may decide that editing mistakes have been made, and wish to discard the new version while saving the original. Or a user might wish to save the new version as a separate document, while saving the original unchanged. Such decisions can be made best at the end of an editing session, when the user knows what has been accomplished, rather than before a session is begun.

COMMENT: During text editing, the computer should always retain a copy of the original document until the user confirms that it should be changed. The original document should **not** be changed automatically as the user enters each editing change.

SEE ALSO: 6.0•18, 6.3•19.

1.4 Data Forms

> *Data forms permit entry of predefined items into labeled fields of specially formatted displays.*

•1 **Combined Entry of Related Data**

In a form-filling dialogue, when a user is entering logically related items, require just one explicit entry action at the end of the transaction sequence, rather than separate entry of each item.

COMMENT: Depending on form design, this practice might involve entering the entire form, or entry by page or section of a longer form. Form design should indicate to users just where explicit entry is required.

COMMENT: Single entry of grouped data will generally permit faster input than item-by-item entry, and should prove more accurate as well. This practice permits user review and possible data correction prior to entry, and also helps the user understand at what point grouped data are processed. It will also permit efficient cross validation of related data items by the computer.

SEE ALSO: 1.0•9, 6.3•6, 6.3•18.

•2 **Flexible Interrupt**

When multiple data items are entered as a single transaction, as in form filling, allow the user to REVIEW, CANCEL, or BACKUP and change any item before taking a final ENTER action.

REFERENCE: BB 2.10; Foley and Wallace, 1974.

SEE ALSO: 1.0•9, 3.3•3 thru •5, 3.5•2, 6.0•10, 6.3•8.

•3 **Minimal Use of Delimiters**

Whenever possible, allow entry of multiple data items without keying special separator or delimiter characters, e.g., hyphens, dollar signs, etc.

EXAMPLE: [See sample displays at the end of this section.]

COMMENT: This can be accomplished either by keying into predefined entry fields or by separating sequentially keyed items with blank spaces. In this context, tabbing from field to field is **not** considered to be keying a special delimiter character.

COMMENT: When data items contain internal blanks, design the entry fields with a predefined structure so that users will not have to key any internal delimiters.

▶ Standard Delimiter Character •4

When a field delimiter **must** be used for data entry, adopt a standard character to be employed consistently for that purpose.

EXAMPLE: A slash (/) may be a good choice.

COMMENT: Choose a special delimiter character that does **not** require shift keying. It may also be necessary to choose a character that does not occur as part of any data entry (except possibly for entry of running text where its occurrence would not be ambiguous).

Data Field Labels •5

For each data field, display an associated label to help users understand what entries can be made.

EXAMPLE:

```
(Good)   Name: _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
         Organization: _ _ / _ _
         Phone: _ _ _ - _ _ _ _
(Bad)    Name, Organization and Phone
         _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
         _ _ / _ _
         _ _ _ - _ _ _ _
```

REFERENCE: BB 2.1.7.

SEE ALSO: 1.0•24, 4.0•11.

▶ Consistent Labeling •6

Make field labels consistent; always employ the same label to indicate the same kind of data entry.

EXAMPLE: A field labeled NAME should always require **name** entry, and not sometimes require something different like **elevation** (cited from an actual system).

EXAMPLE: [See sample displays at the end of this section.]

1.4 Data Forms

•7 ▶ **Protected Labels**

Protect field labels from keyed entry, by making the cursor skip over them automatically when a user is spacing or tabbing.

EXCEPTION: When a user must change a displayed form, including changes to field labels, then that user must be able to override label protection.

REFERENCE: BB 1.8.13; PR 3.3.2, 4.8.1.

SEE ALSO: 1.1•23, 2.0•10, 6.2•5, 6.3•2.

•8 ▶ **Labels Close to Data Fields**

Ensure that labels are sufficiently close to be associated with their proper data fields, but are separated from data fields by at least one space.

REFERENCE: BB 1.9.5; EG 2.3.8.

SEE ALSO: 2.2•9.

•9 ▶ **Standard Symbol for Prompting Entry**

Choose a standard symbol for input prompting and reserve that symbol only for that use.

EXAMPLE: In the examples here, and also in many printed forms, a colon serves to prompt inputs, as

(Good) Time: __________

(Bad) Time __________

COMMENT: Consistent use of a symbol for input prompting in data entry forms, in menus, in command entry lines, etc., will help to cue users that an input is required. A standard symbol used in addition to other formatting cues will help to alert a user to differences between labels and displayed data, between messages requiring input and messages for information only.

REFERENCE: BB 2.5.2.

SEE ALSO: 3.1.3•15, 4.4•10.

Marking Field Boundaries •10

Display special characters or other consistent means of highlighting to clearly delineate each data field.

EXAMPLE: An underscore might be used for this purpose, perhaps broken to indicate the number of symbols required in an entry, as

(Good) `Enter account number: __ __ __ __`

(Bad) `Enter account number:`

EXAMPLE: [See sample displays at the end of this section.]

COMMENT: Such implicit prompts help reduce data entry errors by the user.

REFERENCE: BB 2.2.1; EG 6.3, 6.3.1; MS 5.15.4.3.4; PR 4.8.1; Savage, Habinek and Blackstad, 1982.

SEE ALSO: 1.0•6, 2.2•2, 4.4•15.

▶ Prompting Field Length •11

Provide cues in field delineation to indicate when a fixed or maximum length is specified for a data entry.

EXAMPLE:

(Good) `Enter ID: __ __ __ __ __ __ __ __`

(Bad) `Enter ID (8 characters):`

EXAMPLE: [See sample displays at the end of this section.]

COMMENT: Prompting by delineation is more effective than simply telling the user how long an entry should be. In the example cited here, underscoring gives a direct visual cue as to the number of characters to be entered, and the user does not have to count them.

COMMENT: Similar implicit cues should be provided when data entry is prompted by auditory displays. Tone codes can be used to indicate the type and length of data entries.

REFERENCE: BB 2.2.1; EG 6.3; MS 5.15.4.3.7; PR 4.8.2; Smith and Goodwin, 1970.

SEE ALSO: 4.4•15.

1.4 Data Forms

•12 ▸ **Marking Required and Optional Data Fields**

In designing form displays, distinguish clearly and consistently between required and optional entry fields.

EXAMPLE: Field delineation cues may be used for this purpose, perhaps a broken underscore to indicate required entries and a dotted underscore to indicate optional entries, as

```
License Number: __ __ __ __ __ __
Make: . . . . . . . . . . . . . . .
Year/Model: . . / . . . . . .
```

EXAMPLE: Alternatively, it might be preferable to distinguish required versus optional entry fields by coding their labels, perhaps displaying in parentheses the labels of optional fields.

REFERENCE: BB 2.6; MS 5.15.4.3.4; PR 4.8.6.

SEE ALSO: 4.4•15.

•13 ▸ **Field Markers Not Entered with Data**

When item length is variable, so that a data entry does not completely replace the markers in a field, ignore any remaining field markers in computer processing.

COMMENT: A user should not be expected to erase any field markers. Extra markers should not be processed as part of a data entry if they are not erased.

REFERENCE: BB 2.2.2; EG 6.3.2; MS 5.15.4.3.9; Stewart, 1980.

•14 ▸ **Automatic Justification of Variable-Length Entries**

When item length is variable, provide automatic justification in computer processing; a user should **not** have to justify an entry either right or left.

EXAMPLE: Assuming field delineation by underscore, the following entries should all be considered equivalent

```
Address: 40 Dalton Road________
Address: ________40 Dalton Road
Address: ____40 Dalton Road____
```

COMMENT: If a data entry is shorter than its field length, its position when entered in that field should not matter. The computer can impose its own justification rules when storing and subsequently displaying such data for review.

REFERENCE: BB 2.2.2; EG 6.3.2.

Explicit Tabbing to Data Fields •15

Require users to take explicit keying ("tabbing") action to move from one data entry field to the next; the computer should **not** provide such tabbing automatically.

EXAMPLE: [See sample displays at the end of this section.]

COMMENT: Automatic tabbing may cause cascading of errors, if a skilled typist keys a series of items without looking at the display and has accidentally overrun one of the earlier data fields. An acceptable solution here is to design each field to end with an extra (blank) character space; software should be designed to prevent keying into a blank space, and an auditory signal should be provided to alert the user when that is attempted. This will permit consistent use of tab keying by the user to move accurately from one field to the next, even for touch typists.

REFERENCE: MS 5.15.4.3.6; PR 4.9.1.

SEE ALSO: 1.1•22.

Distinctive Label Format •16

Make labels for data fields distinctive, so that they will not be readily confused with data entries, labeled control options, guidance messages, or other displayed material.

EXAMPLE: Labels might be displayed in capital letters always followed by a colon. Or labels might be displayed in dim characters, with data entries brighter.

EXAMPLE: [See sample displays at the end of this section.]

REFERENCE: BB 2.1.1; MS 5.15.4.3.5; PR 3.3.2.

SEE ALSO: 2.2•8, 4.0•8.

▶ Consistent Label Format •17

When data fields are distributed across a display, adopt a consistent format for relating labels to delineated entry areas.

EXAMPLE: The label might always be to the left of the field; or the label might always be immediately above and left-justified with the beginning of the field.

COMMENT: Such consistent practice will help the user distinguish labels from data in distributed displays.

SEE ALSO: 4.0•7.

•18 ▸ **Label Punctuation as Entry Cue**

The label for each entry field should end with a special symbol, signifying that an entry may be made.

EXAMPLE: A colon is recommended for this purpose, as

Name: __ __ __ __ __ __ __ __ __ __

EXAMPLE: [See sample displays at the end of this section.]

COMMENT: Choose a symbol that can be reserved exclusively for prompting user entries, or at least is rarely used for any other purpose.

REFERENCE: BB 2.5.

SEE ALSO: 4.4•15.

•19 **Informative Labels**

In labeling data fields, employ descriptive wording, or else standard, predefined terms, codes and/or abbreviations; avoid arbitrary codes.

EXAMPLE: Employ descriptive labels such as STANDARD and MODIFIED, rather than abstract codes such as SET A and SET B; MALE and FEMALE, rather than GROUP 1 and GROUP 2.

EXAMPLE:

(Good)

Week: __ Month: __ __ Year: __ __

Social Security No: __ __ __ - __ __ - __ __ __ __

(Bad)

Datecode: __ __ __ __ __

SSAN: __ __ __ __ __ __ __ __ __

EXAMPLE: [See sample displays at the end of this section.]

COMMENT: Do not create **new** jargon. If in doubt, pretest all proposed wording with a sample of qualified users.

REFERENCE: BB 2.1.6; PR 4.5.6.

SEE ALSO: 2.0•12, 4.0•11.

•20 **Data Format Cueing in Labels**

Include in a field label additional cueing of data format when that seems helpful.

EXAMPLE: Date (mm/dd/yy): __ __/__ __/__ __

EXAMPLE: [See sample displays at the end of this section.]

REFERENCE: BB 2.1.8; MS 5.15.4.3.5; PR 4.8.9.

SEE ALSO: 4.0•11.

▸ Labeling Units of Measurement •21

When a measurement unit is consistently associated with a particular data field, include that unit as part of the field label rather than requiring a user to enter it.

EXAMPLE: Cost: $ _ _ _ _

EXAMPLE: Speed (mph): _ _ _

REFERENCE: BB 2.2.6; MS 5.15.4.3.10; PR 4.8.11.

SEE ALSO: 2.2•10, 4.0•11.

▸ Familiar Units of Measurement •22

Employ units of measurement that are familiar to the user.

EXAMPLE:

(Good) Speed Limit: _ _ miles per hour
(Bad) Speed Limit: _ _ feet per second
(Good) Fuel Use: _ _._ miles per gallon
(Bad) Fuel Use: ._ _ gallons per minute

COMMENT: If data must be converted to unfamiliar units, the computer should handle that automatically.

REFERENCE: BB 2.3; MS 5.15.2.1.7.

SEE ALSO: 4.0•17.

▸ Alternative Units of Measurement •23

When alternative measurement units are acceptable, provide space in the data field so that a user can designate the units actually entered.

EXAMPLE: Distance: _ _ _ _ (mi/km) _ _

REFERENCE: MS 5.15.4.3.10; PR 4.8.11.

SEE ALSO: 4.0•11.

1.4 Data Forms

•24 **Form Compatible for Data Entry and Display**

When forms are used for reviewing displayed data as well as for data entry, make the form for data entry compatible with that for display output; use the same item labels and ordering for both.

COMMENT: When a display format optimized for data entry seems unsuited for data display, or vice versa, some compromise format should be designed, taking into account the relative functional importance of data entry and data review in the user's task.

REFERENCE: MS 5.15.3.1.1.a.

SEE ALSO: 2.2•12, 2.5•1, 4.0•6.

•25 ▶ **Form Compatible with Source Documents**

When data entry involves transcription from source documents, ensure that form-filling displays match (or are compatible with) those documents, in terms of item ordering, data grouping, etc.

EXAMPLE: [See sample displays at the end of this section.]

COMMENT: If paper forms are not optimal for data entry, consider revising the layout of the paper form.

COMMENT: If data entries must follow an arbitrary sequence of external information (e.g., keying telephoned reservation data), employ some form of command language dialogue **instead** of form filling, to identify each item as it is entered so that the user does not have to remember and re-order items.

REFERENCE: BB 1.8.9; MS 5.15.3.1.1.b, 5.15.4.3.3; PR 4.8.3, 4.8.5, 4.10.7; Stewart, 1980.

SEE ALSO: 2.5•1, 2.5•14, 3.1.1•4, 4.0•6.

•26 **Minimal Cursor Positioning**

When designing displays for form-filling data entry, minimize user actions required for cursor movement from one field to the next.

COMMENT: Placing all required fields before any optional fields will sometimes make data entry more efficient.

REFERENCE: BB 2.1.3.

SEE ALSO: 1.1•22.

▶ Data Items in Logical Order •27

If no source document or external information is involved, then design forms so that data items are ordered in the sequence in which a user will think of them.

COMMENT: The software designer will need to work with prospective system users to determine what represents a logical sequence of data entries.

REFERENCE: PR 4.8.5.

SEE ALSO: 2.5•14.

Automatic Cursor Placement •28

When a form for data entry is displayed, the computer should place the cursor automatically at the beginning of the first entry field.

EXCEPTION: If a data form is regenerated following an entry error, the cursor should be placed in the first field in which an error has been detected.

REFERENCE: BB 2.1.4; PR 4.9.1.

SEE ALSO: 1.1•21, 3.1.3•29, 4.4•16.

1.4 Data Forms

(Good) **Sample Data Form** **(Good)**

```
                        VISA APPLICATION

NAME: Jones, Andrew David______            VISA: 356 478
      LAST, FIRST MIDDLE

BIRTH COUNTRY: UK    DATE: 3/22/25
                           M  D  Y

   NATIONALITY: UK   PASSPORT: Z196284____

ADDRESS: 5 Fairview Lane______________
         Loughborough, LE11 3RG______
         England______________________

OTHER TRAVELERS ON THIS VISA
                                BIRTH
NAME:                           COUNTRY: DATE:
Jones, Sandra Jean____          UK       10/11/28
Jones, Cynthia Leigh__          FR        6/12/68
________________________        __       __/__/__
________________________        __       __/__/__
LAST, FIRST MIDDLE                       M  D  Y

* Press ENTER when done.
```

These sample displays represent a possible form for entry and review of visa application data. In the good form, data entries are bolded to help distinguish them from labels and field delimiters. Fields are ordered consistently in relation to a (supposed) paper application form, and formatted to facilitate both data entry and data review.

The bad display is annotated to indicate violations of several of the design guidelines proposed here for data forms. The data entries in the bad display were invented to suggest what a user might have produced, if confused by inadequate labeling and the absence of field delimiters.

(Bad) Sample Data Form (Bad)

```
Name Andrew D. Jones              Visa Number 356478

Birthplace London                 Nationality English

Passport Z196284                  Birthdate Mar. 22,

Address  1925
         5 Fairview Lane, Loughborough, L
         E11 3RG, England

Other travelers on this visa
Traveler's Name                   Date of Birth - Place
Sandra J. Jones                        Oct. 11, - 1928
Birmingham
Cynthia L. Jones                       June 12,- 1968
Paris, France█

Press ENTER when done
```

This bad data form display violates in some degree several design guidelines in this section:

1.4•3 Minimal use of delimiters
•6 Consistent labeling
•10 Marking field boundaries
•11 Prompting field length
•15 Explicit tabbing to data fields
•16 Distinctive label format
•18 Label punctuation as entry cue
•19 Informative labels
•20 Data format cueing in labels
•25 Form compatible with source documents

This bad data form also violates various design guidelines pertaining to data display, as noted at the end of Section 2.2.

1.5 Tables

Tables permit data entry and display in row-column format, facilitating comparison of related data sets.

•1 **Tables for Related Data Sets**

When sets of data items must be entered sequentially, in a repetitive series, provide a tabular display format where data sets can be keyed row by row.

EXCEPTION: When the items in each data set exceed the capacity of a single row, tabular entry will usually **not** be desirable, unless there is a simple means for horizontal scrolling.

COMMENT: Row-by-row entry facilitates comparison of related data items, and permits potential use of a DITTO key for easy duplication of repeated entries.

REFERENCE: PR 4.8.4.

SEE ALSO: 2.7.2•4.

•2 **Distinctive Labels**

Design distinctive formats for column headers and row labels, so that users can distinguish them from data entries.

SEE ALSO: 4.0•8.

•3 ▶ **Informative Labels**

Ensure that column headers and row labels are worded informatively, so that they will help guide data entry.

SEE ALSO: 4.0•11.

•4 **Tabbing within Rows**

During tabular data entry, allow users to tab directly from one data field to the next, so that the cursor can move freely back and forth within a row (i.e., across columns).

REFERENCE: MS 5.15.3.8.4.3.

•5 ▶ **Tabbing within Columns**

During tabular data entry, allow users to tab directly from one data field to the next, so that the cursor can move freely up and down a column (i.e., across rows).

REFERENCE: MS 5.15.3.8.4.3.

Automatic Justification of Entries •6

Provide automatic justification of tabular data entries; a user should **not** have to enter blanks or other extraneous formatting characters to achieve proper justification.

EXAMPLE: As a negative example, if a user enters 56 in a field four characters long, the system should **not** interpret 56 __ __ as 5600.

REFERENCE: MS 5.15.2.2.5.

▶ Justification of Numeric Entries •7

Allow users to make numeric entries in tables without concern for justification; the computer should right-justify integers, or else justify with respect to a decimal point if present.

EXAMPLE: A dollars-and-cents entry made at the beginning of a field (14.37 __ __) should automatically be justified to the right (__ __ 14.37) when later displayed.

REFERENCE: PR 4.8.10.

▶ Maintaining Significant Zeros •8

When a user must enter numeric values that will later be displayed, maintain all significant zeros; zeros should not be arbitrarily removed after a decimal point if they affect the meaning of the number in terms of significant digits.

REFERENCE: BB 1.4.3.

SEE ALSO: 2.3•17.

Aiding Entry of Duplicative Data •9

For entry of tabular data, when entries are frequently repeated, provide users with some easy means to copy duplicated data.

EXAMPLE: Perhaps a DITTO key might be provided.

COMMENT: A DITTO capability will speed data entry, and should prove more accurate than requiring users to rekey duplicated data.

1.5 Tables

•10 Row Scanning Cues

For long tables, those with many rows, provide some extra visual cue to help a user scan a row accurately across columns.

EXAMPLE: A blank line might be inserted after every fifth row; or perhaps adding dots between columns in every fifth row might suffice.

EXAMPLE: As an alternative, provide a displayed ruler which a user can move from one row to another.

COMMENT: Visual aids for scanning rows are probably needed more when a user is reviewing and changing displayed data than for initial data entry. Such aids should be provided consistently, however, so that display formats for both data entry and review will be compatible.

SEE ALSO: 2.3•14.

Graphics permit entry of data specially formatted to show spatial, temporal, or other relations among data sets.

Pointing •1

When graphic data entry involves frequent pointing on a display surface, design the user interface so that actions for display control and sequence control are also accomplished by pointing, in order to minimize shifts from one entry device to another.

EXAMPLE: In drawing a flow chart, a user should be able to link predecessor and successor elements directly by pointing at them, or by drawing lines between them, rather than by separately keyed entries.

EXCEPTION: Alphabetic entry for titles, labels, and other annotation of graphic displays will be accomplished more quickly by conventional keyboard input than by pointing.

COMMENT: This recommendation implies extensive use of menus in the margins of a graphic display to permit direct selection of data attributes and control options by pointing. If screen capacity is too limited to permit simultaneous display of both graphic data and menus, then the designer might provide temporary superposition of menu windows on displayed data, or might provide some separate display device to show current options for control entry. Control entry via keyboard and/or function keys will be less satisfactory.

COMMENT: If pointing is performed on some separate input device, such as a stylus on a digitizing tablet, then associated control actions should also be implemented via that device.

COMMENT: For graphics software, a pointing action by a user can accomplish several different logical functions: specifying a displayed element ("pick" function); selecting a system-defined object, attribute or action ("button" or "choice" function); or indicating a location in the conceptual drawing space ("locator" function). A designer must distinguish among these functions, although most users will not.

REFERENCE: Foley and Wallace, 1974; Foley and Van Dam, 1982; Foley, Wallace and Chan, 1984.

SEE ALSO: 1.0•5, and Section 1.1.

1.6 Graphics

•2 **▶ Distinctive Cursor**

Indicate the current cursor position by displaying some distinctive cursor symbol at that point.

COMMENT: The cursor may take various forms on a graphics display. Many designers recommend a plus-sign for this purpose, representing abbreviated cross-hairs whose intersection can mark a position with reasonable precision. In some applications it may help to extend those cross-hairs the full height and width of the display. In some applications it may help to display a cursor incorporating the current values of various attributes (color, size, etc.) that can be selected by a user.

REFERENCE: Foley, Wallace and Chan, 1984.

SEE ALSO: 1.1•1, 1.6•12.

•3 **▶ Easy Cursor Positioning**

Provide users an easy, accurate means of positioning a displayed cursor to point at different display elements and/or display locations.

COMMENT: Cursor positioning is a frequent user action during graphic data entry; an easy means for controlling cursor movement will be essential for efficient performance.

SEE ALSO: 1.1•7.

Confirming Cursor Position •4

For most graphics data entry, pointing should be a dual action, first positioning a cursor at a desired position, and then confirming that position to the computer.

EXCEPTION: An exception to this recommendation would be the freehand drawing of continuous lines ("path specification"), where a computer must store and display a series of cursor positions as they are input by the user; when the user initiates such a line-drawing sequence, a new data point might be recorded automatically whenever the cursor has been moved a certain distance (e.g., 1 mm) or when a certain time has elapsed (e.g., 0.5 s).

COMMENT: During graphics data entry, a cursor will almost always be **somewhere** on the display, but not necessarily at a location intended by the user. In effect, a user needs some way to move the cursor around and some separate action to signal the computer when its position should be recorded.

COMMENT: An interesting case of position confirmation is "rubberbanding", which is a technique to aid line drawing. With rubberbanding, a user can designate the starting point for a line, then move the cursor to various possible end points while the computer continuously shows the line that would result if that end point were confirmed by the user.

REFERENCE: Foley and Wallace, 1974; Foley, Wallace and Chan, 1984.

SEE ALSO: 1.1•4, 1.6.2•2.

Zooming for Precise Positioning •5

When data entry requires exact placement of graphic elements, users should be allowed to request expansion of the critical display area ("zooming") to make the positioning task easier.

REFERENCE: Foley and Wallace, 1974.

SEE ALSO: 1.6.2•11, 2.4•15.

Selecting Graphic Elements •6

Provide users some means for designating and selecting displayed graphic elements for manipulation.

EXAMPLE: Designation might be by pointing, in the case of a discrete element, or might require some sort of outlining action to delineate portions of a complex figure.

SEE ALSO: 1.3•5.

•7 ▸ **Highlighting Selected Elements**

When a user has selected (i.e., pointed at) a displayed graphic element, highlight that element in some way so that the user can anticipate the consequences of any proposed action involving that selection.

EXAMPLE: A dotted border might be displayed around a selected element, or perhaps a selected element might be displayed with video inversion to distinguish it from other elements.

REFERENCE: Foley, Wallace and Chan, 1984.

SEE ALSO: 1.3•7, 2.6•1, 4.2•10.

•8 **Changing Position (Translation)**

When editing graphic data, allow users to reposition selected elements on the display.

COMMENT: Repositioning displayed elements, whether done by "dragging" or "cut-and-paste", will usually prove easier than deleting an element and then recreating it from scratch in the desired location. A capability for moving elements will aid initial data entry as well as any subsequent editing of graphic data.

COMMENT: If an element is moved visibly by dragging across the display, it is probably not necessary to depict it in complete detail in all of its intermediate positions. It might suffice to show it in simplified outline until its new position has been confirmed by the user (or perhaps until it remains in one position for a fixed interval of time), at which point its details could be filled in again by the computer.

SEE ALSO: 1.3•23.

•9 **Deleting Elements**

When editing graphic data, allow users to delete selected elements from the display.

COMMENT: Deletion/erasure will help when mistakes are made during data entry, as well as in any subsequent editing of graphic data. Deletion should be implemented as a reversible action. A general UNDO capability might suffice to reverse deletions. A more extended reversibility might be provided by saving deleted elements in a computer scrap basket from which they can be retrieved any time during a work session in case a deletion is discovered to be a mistake.

Selecting from Displayed Attributes •10

During graphic data entry, allow users to specify attributes for displayed elements — e.g., text font, plotting symbol, line type, color — by selecting from displayed samples illustrating the available options.

EXAMPLE: For line drawing a user might select from displayed samples of thick or thin, solid or broken, etc.

COMMENT: A display of available attributes will serve as a helpful reminder to the user, and will eliminate the need to assign distinctive verbal labels to the various options.

COMMENT: Samples of some attributes may be difficult to display. In complex graphics, for example, specification of line type might involve selection among "brushes", each of which has a "tip" defining the size and shape of the drawing area (a group of pixels) that the user can manipulate. Brushes might have squared tips to draw sharp lines, or rounded tips to draw lines with softer edges. By analogy with artistic painting, a "smear" brush might be provided to average or blend colors along its path. Selective erasure might be accomplished with a brush applying (returning to) the color of the display background.

COMMENT: In most applications, the current selection of data attributes should remain in effect until a new selection is made. In some cases, e.g., following selection of an "erase" attribute, it may help the user if a selected attribute reverts automatically to a default value at the completion of a transaction sequence.

▶ Selecting Colors •11

If users may select colors as an attribute of graphic elements, allow them to specify colors directly by pointing at displayed samples, rather than requiring them to name the colors.

EXCEPTION: If only a few colors are available, their names can probably be used reliably.

COMMENT: If many colors are available, users with normal vision can choose from displayed samples more reliably than from a list of color names. For color-blind users, however, it might be helpful to add names/labels to the displayed samples.

COMMENT: For more elaborate graphic art, it may be helpful to allow users to mix their own colors by sequential selection (i.e., cursor placement), either in a displayed palette or directly in a graphic image. Such color mixing could permit user control of saturation, brightness, and opacity/transparency, as well as hues.

•12 ▸ **Displaying Current Attributes**

During graphic data entry/editing, display the selected attributes that will affect current actions for ready reference by the user.

EXAMPLE: When graphic attributes — plotting symbols, character size, line type, color, etc. — are chosen from displayed menus, it might suffice to highlight the currently selected menu options; alternatively, current selections might be shown in some sort of "reminder" window.

EXAMPLE: A few attributes might be shown by the displayed cursor, i.e., by changing cursor shape, size or color depending upon current attribute selections.

EXAMPLE: If rubberbanding is provided to aid line drawing, then that process itself would show the currently selected line type.

COMMENT: Users may forget what options have been chosen. Displayed reminders will be particularly important in situations where the consequences of a mistaken user action are difficult to reverse, e.g., where it may be hard to erase a wrongly drawn line.

COMMENT: In some applications, display cues may not be adequate to convey attribute information completely. There may not be sufficent room on the display. Or the attributes may derive from underlying models whose characteristics are too complex for simple display representation. In such cases, users should be able to request auxiliary display of such information to determine the operative context for current actions.

SEE ALSO: 1.6.2•2, 3.0•9, 4.0•9, 4.4•13, and Section 3.4.

•13 **Changing Attributes**

When entering or editing graphic data, allow users to change display attributes — e.g., line type, cross-hatching, color — for selected graphic elements.

EXAMPLE: If a figure was created initially with dashed lines, then a user should be able to select the figure, or portions of it, and change the dashed lines to solid lines by specifying that alternative attribute.

COMMENT: If it is easy to change attributes, reversing earlier data entry decisions, then the process of composing graphic displays will be generally easier.

COMMENT: Another approach to changing an attribute might be to rely on general editing capabilities, i.e., to delete the element in question (perhaps using an UNDO command for an element just created) and then redraw it. But a capability for specifying attribute change directly, without element deletion and reentry, will often be helpful.

SEE ALSO: 1.6•6.

▶ Consistent Method for Attribute Selection •14

When editing graphic data, allow users to change display attributes by whatever means were used to select those attributes in the first place.

EXAMPLE: If line type is selected initially from a menu of displayed attributes, then changing a line type should also be accomplished via menu selection.

COMMENT: Many editing changes will be made during data entry, rather than as separate later actions, and thus it is important that entry and editing actions be consistent.

Easy Storage and Retrieval •15

Provide easy means for saving and retrieving graphic displays or their component elements at different stages in their creation.

COMMENT: A user should **not** have to create a graphic image more than once. Once a graphic element has been created, a user should be able to save it for possible re-use.

COMMENT: As a protective measure, a user might wish to save different versions of a graphic display at successive stages during its creation, in order to return to an earlier state if later results seem unsatisfactory. During creation, the elements added to a graphic display can be interrelated in complex ways, and thus stepwise deletion of unwanted elements could prove a difficult process. An UNDO command might be helpful for deleting some of the most recently added elements. But storage and subsequent retrieval of interim versions of the display may be more helpful for a foresighted user.

▶ Naming Displays and Elements •16

Allow users to name graphic displays or designated elements, in order to aid storage and retrieval or manipulation during graphic data entry/editing; and provide means for a user to review a current "catalog" of named elements.

COMMENT: Standard displays and graphic components might be assigned names automatically by the computer, but users will still need a capability to assign their own names to interim versions of displays in creation, or to various elements of those displays. In either case, users may forget what names have been assigned; some "catalog" of currently named elements will serve as a helpful reminder.

COMMENT: For currently displayed material, pointing may be more convenient than naming for the designation of selected elements; but names will certainly aid the retrieval of stored material.

REFERENCE: Gardan and Lucas, 1984.

1.6 Graphics

•17 Automatic Data Registration

Provide automatic registration or alignment of computer-generated graphic data, so that variable data are shown properly with respect to fixed background or map data at any display scale.

COMMENT: When users are required to enter data via some separate device such as a graphics tablet, rather than directly on the display surface, it may be necessary for a user to participate in some computer-prompted procedure for ensuring data registration. Such a procedure may prove error-prone, however, and should be considered an undesirable expedient.

•18 Aids for Entering Hierarchic Data

When graphic data must be entered in an organized hierarchic structure, in different sections and at different levels of increasing detail, provide computer aids for that purpose.

EXAMPLE: For entering map data, a user might have to specify different levels of data storage for a city's name and location, its municipal boundaries, its major road patterns, its street names and house numbers, etc.; computer aids could help that process.

SEE ALSO: 1.0•31, 1.8•12, 2.4•15.

•19 Automatic Data Validation

When graphic data represent relations among real objects, provide appropriate computer logic based on models of physical probability to validate data entries.

EXAMPLE: If data indicate that a military land unit has been reported in the middle of a lake, the computer should call that discrepancy to the user's attention.

COMMENT: If inconsistencies of data entry cannot be resolved immediately, the computer might keep track of unresolved questions pending receipt of further data.

SEE ALSO: 1.7•1.

Plotting data to show their relations in various graphic formats can be aided greatly by appropriate software.

Automated Data Plotting •1

When complex graphic data must be entered quickly, provide computer aids to automate that process.

EXAMPLE: Prestored geographic data and background maps, along with automated entry ("posting") of flight plan data and track data, will permit fast and accurate generation of graphic displays for air traffic control, far beyond the capabilities of manual entry by a user.

COMMENT: Users can create simple graphics or edit stored graphic material fairly quickly, but they can create complex graphic displays only much more slowly. A variety of computer aids can be provided to help enter graphic data. Entry of detailed drawings and/or photographic imagery can be accomplished via a video camera and high-resolution digitizer, perhaps with facilities for a user to edit that process.

Plotting Stored Data •2

Provide automated plotting of computer-stored data at user request, with provision for subsequent editing by a user.

EXAMPLE: A computer might plot the data values from two arrays in a line graph, or three-dimensional data in XYZ coordinates.

COMMENT: In many applications, data intended for graphic display will already be stored in the computer. In such cases a user might specify the graphic format required and edit elements in the resulting display output, without actually having to re-enter the data. When users do have to enter data for graphic display, they might choose form filling or tabular entry for efficiency in the initial input of data and then invoke graphic capabilities for subsequent data editing. In either case, it is important that previously entered data should be accessible for graphic processing.

SEE ALSO: 1.8•7, 1.8•8.

1.6.1 Graphics - Plotting Data

•3 **Predefined Graphic Formats**

When graphic data must be plotted in predefined standard formats, provide templates or skeletal displays for those formats to aid data entry.

EXAMPLE: Sample displays might be stored in the computer to aid in creating standard graphs such as bar graphs, or standard diagrams such as organization charts, or page layouts for typesetting, or maps drawn to different scales or with different projections.

COMMENT: In many applications, it may help to provide flexibility so that general prestored formats can be modified by a user and then saved for subsequent use.

SEE ALSO: 1.6•15.

•4 **Aids for Graph Construction**

When graphs must be constructed for data plotting, provide computer aids for that purpose.

EXAMPLE: Construction aids might include stored templates of different kinds of graphs, prompts to guide users in the definition of scale axes, and aids for format control such as automatic centering of axis labels if requested by a user.

COMMENT: Computer aids for graph construction should be designed to allow flexibility in their use. A user should be allowed to position labels and other graphic elements at will, except where operational requirements may impose fixed formats.

REFERENCE: Foley and Van Dam, 1982.

•5 ▶ **Aids for Scaling**

Provide computer aids to help users specify appropriate scales for graphic data entry.

COMMENT: The computer should handle scaling automatically, subject to review and change by a user. The computer might provide a general template for the plotting scale and prompt the user as necessary to define the scale more exactly, including specification of the origin, linear or logarithmic axes, scale intervals, minimum and maximum values, and labels for axes.

COMMENT: In the process of defining scales the computer might impose rules to ensure that the resulting graphic displays are designed to permit effective information assimilation by their users, e.g., displaying scales with conventional direction, so that numbers increase in value from left to right, or from bottom to top.

REFERENCE: Foley and Wallace, 1974.

SEE ALSO: 2.4.1•1.

Computer Derivation of Graphic Data •6

When graphic data can be derived from data already available in the computer, provide machine aids for that purpose.

EXAMPLE: A computer might fit a smoothed curve through plotted data values, filter out points when drawing a densely defined curve, rescale graphs, invert graphs by exchanging X- and Y-values, convert graphs to show cumulative curves, calculate and display various statistical measures of data distribution, produce a contour plot from gridded data with linear interpolation, plot map contours from latitude-longitude coordinates, calculate bearings, distances, and areas on maps, plot perspective views of objects defined in plan views, plot specified cross-sections of displayed objects, calculate a parts list for a designed assembly, identify critical paths and float time in network scheduling charts, etc.

COMMENT: The machine capacity for generating graphic data by computation will far exceed a user's capabilities in both speed and accuracy.

SEE ALSO: 1.8•8.

1.6.2 Graphics - Drawing

Drawing lines and figures to produce pictorial data of various kinds can be aided greatly by appropriate software.

•1 **Drawing Lines**

When line drawing is required, provide users with aids for drawing straight line segments.

COMMENT: Some applications may require drawing continuous lines freehand.

REFERENCE: Foley and Van Dam, 1982.

•2 **Rubberbanding**

When lines must be drawn at arbitrary positions, lengths and angles, provide a rubberbanding capability, in which the computer displays a tentative line extending from a designated start point to whatever is the currently proposed end point.

COMMENT: This technique permits users to enter or change a line segment rapidly and with confidence by designating its starting point and then simply moving the cursor to the desired end-point, thus placing the "rubberband" line in its intended position. A similar capability should be provided to aid entry/editing of specified outline figures. A rectangle might be rubberbanded by fixing one corner and moving the opposite corner. A circle might be rubberbanded to desired size by fixing its center and changing the extension of its radius.

REFERENCE: Foley and Van Dam, 1982; Foley, Wallace and Chan, 1984.

•3 **Aiding Line Connection**

When line segments must join or intersect, which is true in most drawing, provide computer logic to aid such connection.

COMMENT: An effective computer logic to aid line connection is to provide a so-called "gravity field" surrounding each line segment, so that if a line-drawing cursor is moved within that field the cursor's new line will be extended automatically to intersect the already-displayed line. Note that a "gravity field" need not itself be displayed; users will soon learn to infer its extent by its effect in aiding cursor placement. Because users often seek to join line segments at their ends, it may help to enlarge the zone of attraction at the end of each displayed line to facilitate such end-to-end connection.

COMMENT: The concept of "gravity field" can also be used to align drawn line segments with points in a reference grid, as well as with each other.

REFERENCE: Foley and Van Dam, 1982; Gardan and Lucas, 1984.

Grid Reference for Alignment •4

When graphic elements are created with vertical and horizontal alignment, provide a reference grid that can be requested by a user to aid that alignment.

COMMENT: A reference grid might be displayed merely as a visual aid. In some instances, however, where repeated graphic elements must be aligned in regular fashion, it may be helpful to use a grid to position graphic elements automatically at its intersections. An example might be the construction of organization charts with repeating rows of boxes connected by line segments. "Grid gravity" might be provided automatically during graphic entry, based on "gravity field" connection of drawn lines to grid points, or might be invoked as a separate editing command by a user.

COMMENT: A grid suitable for aiding data entry may not prove equally helpful for subsequent interpretation of data on the completed display. Therefore, after a graphic image has been composed, the user should decide whether or not to include the reference grid in the finished display.

REFERENCE: Foley, Wallace and Chan, 1984.

SEE ALSO: 2.4.1•11.

▶ Changing Grid Intervals •5

When a reference grid is displayed to aid graphic data entry, allow users to change the grid intervals in either or both directions.

COMMENT: For different applications, a user may wish to work with a fine grid or a coarse grid, depending on the quantizing interval of the data being plotted. Some designers recommend a standard grid resolution of 1/20 of graph height or width, but such a standard will not be optimum for every application.

Constraint for Vertical and Horizontal Lines •6

When graphic elements are created with vertical and horizontal lines, allow users to specify appropriate constraints during line drawing.

COMMENT: Here computer logic is invoked to interpret casual freehand gestures by a user as if they were carefully drawn — the electronic equivalent of a draftsman's T-square. Thus a roughly vertical motion by a user could create an exactly vertical line in computer storage and display.

COMMENT: In applications where orthogonal lines predominate, it may be helpful to make constrained drawing the norm, while allowing users to specify free-form drawing as an exception.

REFERENCE: Foley and Van Dam, 1982; Gardan and Lucas, 1984.

1.6.2 Graphics - Drawing

•7

Specifying Line Relations

For precise drawing, allow users to draw lines by specifying their geometric relations with other lines.

EXAMPLE: In computer-aided design, a user might wish to create a new line by declaring it parallel with (or perpendicular to) an existing line.

•8

Drawing Figures

When a user must draw figures, provide computer aids for that purpose.

EXAMPLE: A user might select from a stored set of standard forms — rectangles, circles, etc. — and edit those to create figures or the component elements of figures, rather than having to draw each figure from scratch.

EXAMPLE: Computer logic might be provided to allow a user to create a rectangle simply by designating two opposite corners, or a circle by first specifying its center and then any point on its circumference, with rubberbanding to show the result of any current selection.

COMMENT: Much graphic construction can either be aided in some way (by templates, tracing techniques, grid gravity, etc.), or can employ machine generation of computed or stored forms, often followed by user editing of those forms. A great many different figures can be created by combining simple elements or by specifying geometric parameters (e.g., conic sections). Computer aids that allow such shortcuts can speed figure drawing and make the process more accurate. In some applications, such as constructing organization charts, figures may repeat a number of standard elements. In such cases computer aids can be provided to make the production of figures almost routine.

COMMENT: Some capability for freehand drawing may be needed, particularly in the creation of graphic art, but freehand drawing will not provide sufficient precision for many applications.

REFERENCE: Gardan and Lucas, 1984.

SEE ALSO: 1.6.1•3.

▶ Alternative Methods for Drawing Figures •9

In applications requiring a general capability for drawing figures, provide a choice of methods for specifying graphic elements.

EXAMPLE: A straight line might usually be created by specifying two points, but sometimes it might be easier to specify one point plus a constraint that the line be parallel (perpendicular, tangent) to some other line.

EXAMPLE: A circle might usually be created by specifying its center and a point on its circumference; but sometimes it might be easier to specify a circle by other means — e.g., by two ends of its diameter, or by three points on its circumference, or by its center plus a constraint that it be tangent to some other figure, or by inscribing it within a square.

EXAMPLE: An ellipse might usually be created by specifying two foci and a point on its perimeter, but sometimes it might be easier to specify its center and draw its long and short axes, or it might be inscribed within a rectangle.

EXAMPLE: A regular polygon might usually be created by specifying the end points of one edge and the number of sides, but it also might be specified by its center and one vertex and the number of its sides.

COMMENT: These examples are from the demanding realm of computer-aided design. Simpler kinds of graphic entry may not require such capabilities.

COMMENT: In the use of various figure-drawing aids, it may be helpful if the computer can provide step-by-step prompts for each procedure, e.g., "Now indicate center point", "Now indicate radius", etc.

REFERENCE: Gardan and Lucas, 1984.

Changing Size •10

When editing graphic data, allow users to change the size of any selected element on the display.

COMMENT: Scaling displayed elements to different sizes, expanding or shrinking them, will usually prove easier than deleting an element and then recreating it from scratch in the desired size. A capability for changing the scale of a displayed element will aid initial data entry as well as any subsequent editing of graphic data.

COMMENT: Depending on the application, it may be helpful to provide a continuous sizing capability, or else incremental sizing to various defined scales.

1.6.2 Graphics - Drawing

•11 ▸ **Enlargement for Symbol Drawing**

In applications where users may create special symbols, provide a capability for drawing (or changing) a symbol in large scale, with automatic reduction by the computer to the needed size.

EXAMPLE: Enlargement might aid in specifying shapes to be used for plotting points or for map symbols, or in designing icons or the letters in a font.

COMMENT: When drawing symbols in large scale, a rough sketch may suffice, requiring less dexterity from a user. The desirable degree of scale expansion will depend upon symbol complexity, and can probably be determined by testing. Some designers recommend a 20x20 grid to provide an enlarged pixel representation, on which a user can add or delete pixels to create a symbol.

SEE ALSO: 1.6•5.

•12 **Copying Elements**

Allow users to copy a selected graphic element in order to duplicate it elsewhere or create a repeating pattern.

COMMENT: Many graphic displays contain repeating elements; copying an element already created may prove quicker than redrawing that element from scratch.

COMMENT: In creating patterns, a user will often need to specify a reference point in the original element and then specify where that point should be placed for each copy of that element.

COMMENT: In some special applications, it might help to provide an optional kind of copying capability called "instancing", in which a user can choose to copy a graphic element from a stored template, and then all copies (or instances) will be changed automatically whenever that original template is changed.

SEE ALSO: 1.6•15.

•13 **Rotating Elements**

When editing graphic data that depict objects, allow users to rotate a selected element on the display, in order to show it in different orientations.

COMMENT: Rotation of a displayed element will usually prove easier than deleting an element and then recreating it from scratch in the desired orientation. A capability for rotating an element will aid initial data entry as well as any subsequent editing of graphic data.

SEE ALSO: 2.4.6•5.

Reflection of Elements •14

When users must create symmetric graphic elements, provide a means for specifying a reflection (mirror image) of existing elements.

COMMENT: Many graphic displays contain symmetric figures where if one side has been drawn the other side might be created quickly as a reflected copy of the first, perhaps with some subsequent modification by the user.

COMMENT: Users will need some means for specifying the desired reflection plane, which for practical purposes should probably be constrained to a choice between left-right and up-down reflection.

REFERENCE: Gardan and Lucas, 1984.

Grouping Elements •15

Allow users to designate a group of elements to which graphic editing operations will be applied in common.

EXAMPLE: A user might carefully position two elements with respect to each other, and then wish to move both of them together while preserving their relative positions.

COMMENT: Grouping elements might be a temporary action, intended for just a few successive editing operations, or it might be specified more permanently via some sort of "make group" command.

SEE ALSO: 1.6•6.

Merging Elements •16

In the special case when a drawn object can be created by the junction or disjunction of other graphic elements, provide computer aids for merging those elements by boolean combination.

EXAMPLE: In showing the junction of two objects comprising the components of some more complex object, a computer might calculate and draw their intersection, automatically dealing with overlapped data sets and concealed contours.

COMMENT: This technique can represent the intersection of solid objects and also the result of drilling holes in an object.

REFERENCE: Gardan and Lucas, 1984.

1.6.2 Graphics - Drawing

•17 **Filling Enclosed Areas**

When area coding is required, provide aids to allow users to fill an enclosed area with a selected attribute (color, shading or cross-hatching) by a simple specification action, rather than by having to trace over the area involved.

EXAMPLE: For many applications, it may suffice if a user can simply point at one of several displayed attributes (color patches, brightness levels, hatching patterns) and then point at the area to be filled.

COMMENT: A user might wish to shade the bars of a bar chart, or the wedges in a pie chart, or the various components of a drawn diagram or picture.

•18 **Automatic Figure Completion**

In applications where design rules have been previously defined, provide computer aids to complete automatically any details of graphic data entry covered by those rules.

EXAMPLE: The computer might automatically add dimensional annotation to drafted figures.

EXAMPLE: When drawing or editing a polygon, the computer might automatically maintain closure if additional vertices are specified, rather than requiring the user to close the figure manually.

EXAMPLE: In computer-aided design, if the flanges of connected components are designed with arcs of standard radius, then a user might draw those joints square and ask the computer to round them.

EXAMPLE: A computer might create perspective drawings automatically from plan and elevation data, with hidden parts eliminated.

EXAMPLE: In drawing flow charts, a computer might automatically add the arrow to a connecting line, depending upon the direction in which the line was drawn (or the sequence in which its points were designated).

REFERENCE: Gardan and Lucas, 1984.

SEE ALSO: 1.6.1•6.

Stored Models •19

When drawings are variations on a common theme, consider providing a computer model that will allow users to create particular instances by entering appropriate parameters.

EXAMPLE: An aerodynamic model might be invoked to help create (and evaluate) an aircraft wing design.

EXAMPLE: For designing a workplace for human use, it might be helpful to store a body model from which the computer could draw automatically a sample user of any specified size percentile, and then move body parts of the displayed sample user to ensure that all controls are within reach.

COMMENT: Different kinds of models might be needed, including models based on geometric, surface, and solid relations, as well as even more complex logical models.

REFERENCE: Foley and Van Dam, 1982; Gardan and Lucas, 1984.

1.7 Data Validation

Data validation refers to checking entries for correct content and/or format, as defined by software logic.

•1 **Automatic Data Validation**

Provide software for automatic data validation to check any item whose entry and/or correct format or content is required for subsequent data processing.

EXAMPLE: If a date is entered as "February 31", the computer should generate an error message asking for a revised entry.

COMMENT: Do not rely on a user always to make correct entries. Computer aids for checking data entries will improve accuracy.

COMMENT: Some data entries, of course, may not need checking, or may not be susceptible to computer checking, such as free text entries in a COMMENT field.

REFERENCE: MS 5.15.2.1.5; PR 4.12.4.

SEE ALSO: 6.3•17, 6.3•18.

•2 **Accepting Correct Entries**

Ensure that every possible correct data entry will be accepted and processed properly by the computer.

EXAMPLE: As a negative example, on 1 June 1983, after several previous months of successful use, the computers controlling Massachusetts automobile emission inspections failed; it was discovered that they would **not** accept a "June" entry.

COMMENT: This guideline states the obvious, and might seem unnecessary except for occasional design lapses such as that cited in the example.

•3 **Non-Disruptive Error Messages**

If data validation detects a probable error, display an error message to the user at the completion of data entry; do **not** interrupt an ongoing transaction.

SEE ALSO: 4.3•10.

Deferral of Required Data Entry •4

If a user wishes to defer entry of a required data item, require the user to enter a special symbol in the data field to indicate that the item has been temporarily omitted rather than ignored.

REFERENCE: MS 5.15.4.3.11; PR 4.8.7, 4.12.2.

SEE ALSO: 4.0•2.

▶ Reminder of Deferred Entry •5

If a user has deferred entry of required data but then requests processing of entries, signal that omission to the user and allow immediate entry of missing items or perhaps further deferral.

REFERENCE: BB 5.2.4; MS 5.15.4.3.11, 5.15.7.5.c; PR 4.8.7.

Timely Validation of Sequential Transactions •6

In a repetitive data entry task, validate the data for one transaction and allow the user to correct errors before beginning another transaction.

COMMENT: This is particularly important when the task requires transcription from source documents, so that a user can detect and correct entry errors while the relevant document is still at hand.

SEE ALSO: 3.5•12, 6.3•9.

Optional Item-by-Item Validation •7

For novice users, consider providing optional item-by-item data validation within a multiple-entry transaction.

COMMENT: This capability, which might be termed an "interim ENTER", may sometimes help a novice user who is uncertain about the requirements imposed on each data item. But item-by-item processing may slow skilled users. Providing such a capability as an optional feature would help novices without hindering more experienced users.

REFERENCE: EG 6.3.9, 7.1.

SEE ALSO: 4.3•10.

1.8 Other Data Processing

Other data processing aids may be provided to facilitate data entry.

•1 **Default Values**

When likely default values can be defined for the data entries in a particular task, offer those default values to speed data entry.

•2 **▶ Defaults for Sequential Entries**

If a series of default values have been defined for a data entry sequence, allow a user to default all entries or to default until the next required entry.

COMMENT: Where a set of default values has been defined, a user may not wish to specify that each default value should be accepted for each data field individually. It might be quicker to accept the set of defaults by a single action.

•3 **▶ User Definition of Default Values**

When interface designers cannot predict what default values will be helpful, permit users (or perhaps a system administrator) to define, change or remove default values for any data entry field.

REFERENCE: MS 5.15.6.8.

•4 **▶ Display of Default Values**

On initiation of a data entry transaction, display currently defined default values in their appropriate data fields.

COMMENT: Do not expect users to remember them.

COMMENT: It may be helpful to mark default values in some way to distinguish them from new data entries.

REFERENCE: BB 2.1.10; MS 5.15.6.7.

SEE ALSO: 4.4•7, 6.3•15.

▶ Easy Confirmation to Enter Default Values •5

Provide users with some simple means to confirm acceptance of a displayed default value for entry.

EXAMPLE: Simply tabbing past the default field may suffice.

COMMENT: Employ similar techniques when a user must review the accuracy of previously entered data.

REFERENCE: MS 5.15.6.7, 5.15.6.10.

SEE ALSO: 6.3•15.

▶ Temporary Replacement of Default Values •6

Allow users to replace any data entry default value with a different entry, without thereby changing the default definition for subsequent transactions.

REFERENCE: MS 5.15.6.9.

Automatic Generation of Routine Data •7

For routine data that can be derived from existing computer records, program the computer to access and enter such data automatically.

EXAMPLE: As a negative example, do **not** require a user to identify a work station in order to initiate a transaction, nor to include other routine data such as current date and transaction sequence codes.

EXCEPTION: Some data entry routines may be imposed in the interest of security, but at the risk of hindering a user in achieving effective task performance. Other means of ensuring data security should be considered.

REFERENCE: BB 2.4.2; MS 5.15.2.1.6.

SEE ALSO: 6.1•1, 6.3•14.

Automatic Computation of Derived Data •8

Provide automatic computation of derived data, so that a user does not have to calculate and enter any number that can be derived from data already accessible to the computer.

EXAMPLE: Statistical descriptors such as sums, means, etc., can all be derived automatically by appropriate software.

REFERENCE: MS 5.15.2.1.6.

SEE ALSO: 6.3•14.

1.8 Other Data Processing

•9 User Review of Prior Entries

When data entries made in one transaction are relevant to a subsequent transaction, program the computer to retrieve and display them for user review rather than requiring re-entry of those data.

REFERENCE: BB 2.4.2.

SEE ALSO: 1.0•1, 6.3•13.

•10 Automatic Entry of Redundant Data

If data are accessible to the computer that are logically related to other entries, program the computer to retrieve and enter those redundant data items automatically.

EXAMPLE: As a negative example, a user should **not** have to enter both an item name and identification code when either one defines the other.

EXCEPTION: Redundant entry may be needed for resolving ambiguous entries, for user training, or for security (e.g., user identification).

COMMENT: When verification of previously entered data is required, ask users to review and confirm data items rather than re-enter them.

REFERENCE: BB 2.4.2, 4.3.6; EG 6.3.10; MS 5.15.2.1.6.

SEE ALSO: 6.3•14.

•11 Automatic Cross-File Updating

Provide automatic cross-file updating whenever necessary, so that a user does not have to enter the same data twice.

EXAMPLE: If an aircraft has been assigned to a mission, the computer should automatically update both aircraft status and mission assignment files to indicate that commitment.

REFERENCE: MS 5.15.2.1.6.

SEE ALSO: 1.0•1, 6.3•14.

Aids for Entering Hierarchic Data •12

When data must be entered in an organized hierarchic structure, in different sections and at different levels of increasing detail, provide computer aids for that purpose.

COMMENT: At the least, the computer should provide the user a schematic summary display of any defined data structure for general orientation, with its branches and levels labeled for convenient reference. When a user specifies any portion of the structure for data entry or editing, the computer should display that section of data appropriately labeled, and perhaps show in the display margin a diagram indicating what portion of the overall data structure is currently being displayed.

COMMENT: When data at one level in a hierarchy are dependent on data entries at other (usually subordinate) levels, the computer should handle cross-level bookkeeping automatically, just as for cross-file updating.

COMMENT: For entering hierarchic data, a user must specify where in the data structure any new data should be added. If the data structure is complex, it may help if the computer automatically prompts the user to make the appropriate data entries at different levels.

COMMENT: If a user may need to change the data structure, then computer aids may be needed for that purpose as well as for data entry. The computer should bookkeep automatically any changing relations among the data in different sections that might result from changes to the overall data structure.

SEE ALSO: 1.0•31, 1.6•18, 2.4.8•11.

1.9 Design Change

Design change of software supporting data entry functions may be needed to meet changing operational requirements.

•1 **Flexible Design for Data Entry**

When data entry requirements may change, which is often the case, provide some means for users (or a system administrator) to make necessary changes to data entry functions.

COMMENT: Data entry functions that may need to be changed are those represented in these guidelines, including changes to procedures, entry formats, data validation logic, and other associated data processing.

COMMENT: Many of the preceding guidelines in this section imply a need for design flexibility. Much of that needed flexibility can be provided in initial interface design. Some guidelines, however, suggest a possible need for subsequent design change, and those guidelines are cited below.

SEE ALSO: 1.3•22, 1.4•25, 1.7•1, 1.8•3, 1.8•8, 1.8•10, 1.8•11.

Section 2
DATA DISPLAY

DATA DISPLAY

Data display refers to computer output of data to a user, and assimilation of information from such outputs. Some kind of display output is needed for all information handling tasks. Data display is particularly critical in monitoring and control tasks. Data may be output on electronic displays, or hardcopy printouts, or other auxiliary displays and signaling devices including voice output, which may alert users to unusual conditions.

In this discussion, data are considered to be display elements related to a user's information handling task. Displayed data might consist of stock market quotations, or the current position of monitored aircraft, or a page of text, or a message from another user. Displayed data might provide guidance to a user in performing a maintenance task, or might provide instruction to a user who is trying to learn mathematics or history.

There might be some display elements that themselves do not constitute task-related data. Those elements include labels, prompts, computer-generated advisory messages and other guidance that helps a user interact with a computer system. Although such user guidance display features are sometimes mentioned here in connection with data display, they are discussed more extensively in Section 4 of these guidelines.

In general, somewhat less is known about data display, and information assimilation by the user, than about data entry. In current information system design, display formatting is an art. Guidelines are surely needed. But these guidelines may simply serve to help a designer become more proficient in the art.

It must be recognized that guidelines cannot tell a designer what the specific contents of a display should be, but only how those contents should be presented. The specific data that must be displayed can only be determined through a careful task analysis to define the user's information requirements.

For effective task performance, displayed data must be relevant to a user's needs. An early statement of the need for relevance in data display, although written before common adoption of gender-free wording, otherwise seems valid still:

> When we examine the process of man-computer communication from the human point of view, it is useful to make explicit a distinction which might be described as contrasting "information" with "data." Used in this sense, information can be regarded as the answer to a question, whereas data are the raw materials from which information is extracted. A man's questions may be vague, such as, "What's going on here?" or "What should I do now?" Or they may be much more specific. But if the data presented to him are not relevant to some explicit or implicit question, they will be meaningless. . . .
>
> What the computer can actually provide the man are displays of data. What information he is able to extract from those displays is indicated by his responses. How effectively the data are processed, organized, and arranged prior to

presentation will determine how effectively he can and will extract the information he requires from his display. Too frequently these two terms **data** and **information** are confused, and the statement, "I need more information," is assumed to mean, "I want more symbols." The reason for the statement, usually, is that the required information is not being extracted from the data. Unless the confusion between data and information is removed, attempts to increase information in a display are directed at obtaining more data, and the trouble is exaggerated rather than relieved.

(Smith, 1963b, pages 296-297)

Certainly this distinction between data and information should be familiar to psychologists, who must customarily distinguish between a physical stimulus (e.g., "intensity" of a light) and its perceived effect ("brightness"). The distinction is not familiar to system designers, however, although the issue itself is often addressed. In the following description of what has been called the "information explosion", notice how the terms data and information are used interchangeably, confounding an otherwise incisive and lively analysis:

The sum total of human knowledge changed very slowly prior to the relatively recent beginnings of scientific thought. But it has been estimated that by 1800 it was doubling every 50 years; by 1950, doubling every 10 years; and by 1970, doubling every 5 years. . . . This is a much greater growth rate than an exponential increase. In many fields, even one as old as medicine, more reports have been written in the last 20 years than in all prior human history. And now the use of the computer vastly multiplies the rate at which information can be generated. The weight of the drawings of a jet plane is greater than the weight of the plane. The computer files of current IBM customer orders contain more than 100 billion bits of information — more than the information in a library of 50,000 books.

For man, this is a hostile environment. His mind could no more cope with this deluge of data, than his body could cope with outer space. He needs protection. The computer — in part the cause of the problem — is also the solution to the problem. The computer will insulate man from the raging torrents of information that are descending upon him.

The information of the computerized society will be gathered, indexed, and stored in vast data banks by the computers. When man needs a small item of information he will request it from the computers. The machines, to satisfy his need, will sometimes carry on a simple dialogue with him until he obtains the data he wants. With the early computers, a manager would often have dumped on his desk an indigestible printout — sometimes several hundred pages long. Now the manager is more likely to request information when he needs it, and receive data about a single item or situation on a screen or small printer.

It is as though man were surviving in the depths of this sea of information in a bathyscaphe. Life in the bathyscaphe is simple, protected as it is from the pressure of the vast quantities of data. Every now and then man peers through the windows of the bathyscaphe to obtain facts that are necessary for some purpose or other. The facts that he obtains at any one time are no more than his animal brain can handle. The information windows must be designed so that man, with his limited capabilities, can locate the data he wants and obtain simple answers to questions that may need complex processing.

(Martin, 1973, page 6)

Some experts understand this distinction clearly, such as Hannemyr and Innocent (1985) who write about 'transforming . . . data into information'. But many system designers and users still fail to recognize the difference.

Once a designer has determined what data must be displayed, through analysis of user information requirements, the next step is to decide how those data might best be formatted. Data might be displayed as text, or in data forms, tables and/or various graphic formats. Each of those types of data display is considered separately in the guidelines presented here.

In some applications, the nature of the data will dictate the necessary format, as in the graphic situation displays used for air traffic control. In some applications, equipment limitations may constrain display formatting, as in systems without graphic capability. In many applications, however, a designer will have considerable latitude in choosing how to display data. Good judgment may be needed to decide when pictures or diagrams should be displayed rather than narrative text, or vice versa.

In the subsections of guidelines dealing with text, or data forms, or tables, or the various types of graphic displays, the initial guidelines describe generally the circumstances in which that particular type of data display may be appropriate. The design decision will require careful analysis of the users' information handling tasks to determine just what circumstances will actually prevail.

For data display, as in other areas of user interface design, some general concepts deserve emphasis, including the importance of context, consistency, and flexibility. Somehow a means must be found to provide and maintain context in data displays so that a user can find needed information. Task analysis may point the way here, indicating what data are relevant to each step in task performance. Design guidelines must emphasize the value of displaying no more data than the user needs, and the importance of maintaining consistent display formats so that the user always knows where to look for different kinds of information, on any one display and from one display to another.

Detailed user information requirements will vary, however, and may not be completely predictable in advance, even with careful task analysis. Thus flexibility is needed so that a user can tailor data displays on line to meet current needs. Such flexibility is sometimes provided through optional data category selection and display offset and expansion features. If options for tailoring display coverage are provided, a user can adjust the assimilation of displayed data in a way analogous to the self pacing of data entry.

When a user must both enter and retrieve data, which is usually the case, the formatting of data displays should be consistent with the methods used for data entry. As an example, if data entry is accomplished via form filling, with specially formatted data fields, subsequent retrieval of that data set should produce an output display with the same format, especially if the user is expected to make changes to the displayed data or additional entries. Where compaction of data output is required for greater efficiency, perhaps to review multiple data sets in a single display frame, the displayed items should retain at least the same ordering and labeling as when those fields were used for data entry.

Introduction

Display design must also take into account the type of dialogue used for sequence control, and with hardware capabilities. Where user inputs are made via menu selection, using a pointing device like a lightpen, then display formats should give prominence (and adequate separation) to the labeled, lightpennable options. Location of multifunction keys at the display margin, to be labeled on the adjacent portion of the display itself, may provide flexibility for both data entry and sequence control, but will necessarily constrain display formatting.

These general concepts underlie many of the guidelines for data display proposed in the following pages. As for the other areas of user interface design, an attempt has been made to write guidelines for data display in functional terms, insofar as possible without reference to specific display devices and questions of hardware implementation. As a practical matter, however, available display technology will inevitably influence the wording of guidelines.

A discerning reader will note that the guidelines presented here deal almost exclusively with visual displays; there are only a few references to other possible display modes. Moreover, most of these guidelines implicitly assume a fairly large visual display, with room to show different kinds of data at one time — in effect, a display with about 24 lines of 80 characters, much like the devices we now use. Consequently, many of these guidelines will not apply in applications where displays are constrained to a smaller size, such as "briefcase" terminals or handheld display devices.

As display technology develops further, it seems inevitable that some of the guidelines proposed here must be reconsidered, and other guidelines added. As an example, we may anticipate increased use of graphics in future information display design, with moving (and talking) pictures such as those we now enjoy in displays designed for entertainment.

The guidelines proposed here are intended for display designers. If we regard displays as contrived arrangements of data, then the guidelines refer to that contrivance. What happens in applications where the computer provides a flexible capability allowing users to contrive many of their own displays? If a user composes a poor page of text, with long sentences, flawed grammar, inconsistent spacing, etc., can a software designer be held responsible? Presumably not, or at least not today.

One might imagine future systems, however, where some form of expertise is stored in the computer, including expertise on user interface design. In applications where users design their own displays, a computer might someday suggest pertinent guidelines, or perhaps even enforce design rules where warranted. For example, if a user entered irregularly spaced text, a smart computer might regularize the spacing in subsequent output of that text.

The guidelines presented here can themselves be regarded as a long multipage data display. Problems of display organization arise in presenting the guidelines material, in terms of content, wording, and format. As in other sections, the topical organization of these guidelines is based on function, dealing with different types of displayed data and display manipulation. As in other sections, the guidelines here recommend specific ways to accomplish some very general objectives.

Introduction

Objectives:

Consistency of data display

Efficient information assimilation by the user

Minimal memory load on user

Compatibility of data display with data entry

Flexibility for user control of data display

Functions

Data display refers to computer output of data to a user, and assimilation of information from such outputs.

Necessary Data Displayed •1

Ensure that whatever data a user needs for any transaction will be available for display.

EXAMPLE: As a minor example, header information should be retained or generated anew when a user is paging/scrolling data tables.

EXAMPLE: As a negative example, even temporary loss of needed data, as might be caused by display blanking during automatic data update, is **not** acceptable in many design applications.

COMMENT: The designer of user interface software must employ some method of task analysis (e.g., operational sequence diagrams) in order to determine a user's detailed information requirements for any transaction.

COMMENT: If data requirements exceed a user's ability to assimilate information from the display, break the task into smaller steps. Prototype testing may be required to determine optimum data displays for critical tasks.

COMMENT: A user should not have to remember data from one display to the next.

REFERENCE: BB 4.3.6; EG 2.3.15; Stewart, 1980; Tullis, 1983.

SEE ALSO: 4.0•5.

2.0 General

•2

▸ Only Necessary Data Displayed

Tailor displayed data to user needs, providing only necessary and immediately usable data for any transaction; do **not** overload displays with extraneous data.

EXAMPLE:

(Good)

CODE	DATA TYPE
su =	Summary
d =	Detailed list
se =	Sequences

(Bad)

CODE	DATA TYPE	IMPLEMENTED
su =	Summary	5-17-82
d =	Detailed list	7-14-82
se =	Sequences	9-25-82

COMMENT: Display of extraneous data may confuse a user and hinder assimilation of needed information.

COMMENT: When user information requirements cannot be exactly anticipated by the designer, allow users to tailor displays on line by controlling data selection (Section 2.7.1), data coverage within a display frame (Section 2.7.2), suppression of displayed data (Section 2.7.4), and data window overlay (Section 2.7.5).

REFERENCE: BB 1.7, 1.8.10; EG 3.1.4, 3.3.1; MS 5.15.3.1.2, 5.15.4.6.2; Stewart, 1980; Tullis, 1981.

SEE ALSO: 2.0•8, 2.7•1, 2.8•1, 4.0•5.

Data Displayed in Usable Form •3

Display data to users in directly usable form; do **not** make users convert displayed data.

EXAMPLE: If altitude might be required in either meters or feet, then display **both** values.

EXAMPLE: This recommendation applies to error messages and other forms of user guidance as well as to data displays.

(Probably adequate)	`Character in NAME entry cannot be recognized.`
(Too cryptic)	`Error 459 in column 64.`

COMMENT: Do **not** require a user to transpose, compute, interpolate, or translate displayed data into other units, or refer to documentation to determine the meaning of displayed data.

REFERENCE: BB 3.3; EG 3.3.4; MS 5.15.3.1.3.

SEE ALSO: 4.4•1.

Data Display Consistent with User Conventions •4

Display data consistently with standards and conventions familiar to users.

EXAMPLE: As a negative example, if users work with metric units of measurement, do **not** display data in English units.

EXAMPLE: Computer time records that are not in directly usable format should be converted for display, to a conventional 12-hour (AM/PM) clock or a 24-hour clock, in local time or whatever other time standard is appropriate to user needs.

EXAMPLE: Calendar formats should follow user customs.

(American calendar)

S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

(European calendar)

S	1	8	15	22	29
M	2	9	16	23	30
T	3	10	17	24	31
W	4	11	18	25	
T	5	12	19	26	
F	6	13	20	27	
S	7	14	21	28	

REFERENCE: BB 3.4; EG 2.2.4.

SEE ALSO: 4.0•16.

▶ Establishing Display Standards •5

When no specific user conventions have been established, adopt some consistent interface design standards for data display.

•6 **Consistent Display Format**

For any particular type of data display, maintain consistent format from one display to another.

COMMENT: When an item is missing from a standard format, display that item as a labeled blank rather than omitting it altogether.

REFERENCE: BB 1.1.1; MS 5.15.3.2.1; Stewart, 1980.

SEE ALSO: 4.0•6.

•7 **Display Consistent with Entry Requirements**

Ensure that data display is consistent in word choice, format, and basic style with requirements for data and control entry.

EXAMPLE: When the computer displays a list of current files, the names in that list should be in a format which would be recognized by the computer if they were part of a control entry; thus a user could mimic the displayed list if specifying a file for editing or mailing.

COMMENT: When composing data and control entries, users will tend to mimic the vocabulary, formats, and word order used in computer displays, including displayed data, labels, error messages, and other forms of user guidance. When entry formats are consistent with display formats, users are more likely to compose an accceptable entry on their first try.

REFERENCE: Good, Whiteside, Wixon and Jones, 1984; Mooers, 1983; Zoltan-Ford, 1984.

SEE ALSO: 3.0•13, 4.0•18.

•8 **User Control of Data Display**

Allow users to control the amount, format, and complexity of displayed data as necessary to meet task requirements.

COMMENT: An experienced user may be able to deal with more complex displays than a novice. But a user experienced in one task may be a novice in another. Thus a range of display tailoring capabilities may be desirable for any particular task.

COMMENT: Increasing the options for user control of data displays will complicate what a new user must learn about a system, and so will involve a trade-off against simplicity of user interface design.

REFERENCE: EG 3.4.2.

SEE ALSO: 2.0•2, 2.8•1, and Section 2.7.

▶ User Changes to Displayed Data •9

Allow users to change displayed data or enter new data when that is required by a task.

COMMENT: For some displays, of course, it is not desirable for users to change data, such as in operations monitoring (process control) displays, or displays permitting access to a protected data base.

COMMENT: Some consistent formatting cue, perhaps different cursor shape or different initial cursor placement, should be provided to inform users when displayed data can or cannot be changed.

REFERENCE: PR 4.4.

SEE ALSO: 1.0•6, 1.3•2, 6.2•4.

▶ Protection of Displayed Data •10

When protection of displayed data is essential, maintain computer control over the display and do **not** permit a user to change controlled items.

COMMENT: Never assume compliance with instructions by the user, who may attempt unwanted changes by mistake, or for curiosity, or to subvert the system.

REFERENCE: EG 3.4.8.

SEE ALSO: 1.1•23, 1.4•7, 6.2•3, 6.3•2.

Context for Displayed Data •11

Ensure that each data display will provide needed context, recapitulating prior data as necessary so that a user does not have to rely on memory to interpret new data.

COMMENT: When user information requirements cannot be determined in advance, it may be desirable to provide a separate display window as a "notepad" in which a user can preserve needed items by marking those to be saved.

COMMENT: If data **must** be remembered from one display to another, display no more than four to six items.

REFERENCE: BB 4.3.6; EG 2.3.15.

Familiar Wording •12

The wording of displayed data and labels should incorporate familiar terms and the task-oriented jargon of the users, and avoid the **un**familiar jargon of designers and programmers.

COMMENT: When in doubt, pretest the meaning of words for prospective users to ensure that there is no ambiguity.

REFERENCE: BB 3.7.1, 3.7.4; EG 3.4.5, 4.2.13; PR 4.5.6.

SEE ALSO: 1.4•19, 4.0•16, 4.0•17, 4.3•3.

•13 **▶ Consistent Wording**

For displayed data and labels, choose words carefully and then use them consistently.

EXAMPLE: (Good)
```
Record A Change
Record B Change
Record C Change
```
(Bad)
```
Update of Record A
Record B Maintenance
Change in Record C
```

EXAMPLE: As a negative example, the word "screen" should not be used to mean "display frame" in one place, and "menu selection option" in another.

COMMENT: Consistent word usage is particularly important for technical terms. Standard terminology should be defined and documented in a glossary for reference by interface designers as well as by users.

REFERENCE: BB 1.2.2, 3.7.2; EG 3.4.5, 4.2.13; Pakin and Wray, 1982.

•14 **▶ Consistent Wording Across Displays**

Ensure that wording is consistent from one display to another.

EXAMPLE: The title of a display should be identical to the menu option used to request that display.

REFERENCE: BB 3.7.4.

•15 **▶ Consistent Grammatical Structure**

Use consistent grammatical structure for data and labels within and across displays.

EXAMPLE: (Good)
```
Starting date:
Leaving date:
Home phone:
Work phone:
```
(Bad)
```
Starting date:
When did you quit:
Home phone:
Phone number at work:
```

COMMENT: Even minor inconsistencies can distract a user and delay comprehension as the user wonders momentarily whether some apparent difference represents a **real** difference.

REFERENCE: Pakin and Wray, 1982.

SEE ALSO: 4.0•23.

Minimal Use of Abbreviation •16

Display complete words in preference to abbreviations.

EXCEPTION: Abbreviations may be displayed if they are significantly shorter, save needed space, and will be understood by the prospective users.

EXCEPTION: When abbreviations are used for data entry, then corresponding use of those abbreviations in data display may help a user learn them for data entry.

REFERENCE: BB 3.1, 3.1.1, 3.1.5; EG 4.1.3; MS 5.15.3.2.3.

▶ Common Abbreviations •17

When abbreviations are used, choose those abbreviations that are commonly recognized, and do not abbreviate words that produce uncommon or ambiguous abbreviations.

EXAMPLE: In a process control application where system components are commonly abbreviated, messages to users could include those common abbreviations, while displaying in full form those words that are not commonly abbreviated, as

(Acceptable)	CST pressure low
(Poor)	Condensate Storage Tank prssr lw
(Acceptable)	Restricted Acct
(Poor)	Rstr Account

COMMENT: The point here is that when abbreviation is necessary due to space constraints, often a designer can still choose which words will be abbreviated. The words chosen for abbreviation should be those that are commonly known in their abbreviated form, and/or those words whose abbreviations can be unambiguously interpreted.

REFERENCE: BB 3.1.6.

▶ Simple Abbreviation Rule •18

When defining abbreviations, follow some simple rule and ensure that users understand that rule.

COMMENT: Abbreviation by truncation is the best choice, except when word endings convey inportant information. When a truncation rule is used, abbreviations are easy for a designer to derive and easy for a user to decode.

COMMENT: If an abbreviation deviates from the consistent rule, it may be helpful to give it some special mark whenever it is displayed.

REFERENCE: BB 3.1.2; MS 5.15.3.2.3; PR 4.5.6; Moses and Ehrenreich, 1981; Rogers and Moeller, 1984.

SEE ALSO: 1.0•17 thru 1.0•23.

•19 **▸ Distinctive Abbreviations**

Ensure that abbreviations are distinctive, so that abbreviations for different words are distinguishable.

REFERENCE: BB 3.1; MS 5.15.3.2.3; Moses and Ehrenreich, 1981.

•20 **▸ Minimal Punctuation of Abbreviations**

Minimize punctuation of abbreviations and acronyms.

EXAMPLE: (Good) USAF
(Bad) U.S.A.F.

EXCEPTION: Punctuation should be retained when needed for clarity, e.g., "4-in. front dimension" rather than "4 in front dimension".

EXCEPTION: Punctuation of abbreviations might be justified when an abbreviation represents more than one word, and more than the first letter of each word is included in the abbreviation, e.g., "common services" abbreviated as "COM.SER" rather than "COMSER".

REFERENCE: BB 1.3.5; EG 2.2.14; MS 5.15.3.2.3.

•21 **▸ Dictionary of Abbreviations**

If abbreviations **are** used, provide a dictionary of abbreviations available for on-line user reference.

REFERENCE: BB 3.1.3; MS 5.15.3.2.3.

SEE ALSO: 4.4•20.

Text displays provide output of stored textual data, along with messages and other text intended for user guidance.

Conventional Text Display •1

Ensure that computer-generated displays of textual data, messages, or instructions, will generally follow design conventions for printed text.

EXAMPLE: [See sample displays at the end of this section.]

COMMENT: Adoption of familiar design conventions for text display will permit users to rely on prior reading skills.

Printing Lengthy Text Displays •2

When a user must read lengthy textual material, consider providing that text in printed form rather than requiring the user to read it on-line.

COMMENT: Reading lengthy text on an electronic display may be 20-30 percent slower than reading it from a printed copy.

COMMENT: There are many good reasons for displaying lengthy textual material on line. Lengthy text may be displayed for editing, mailing, or search tasks. Or a lengthy text might be updated frequently, and so on-line display would be the best way to ensure that all users are reading the most recent version. The intent of this guideline is not to discourage such on-line display of text when that is needed, but rather to discourage on-line display when the text would be more useful in paper form. For instance, if HELP displays consist merely of screen after screen of text which is not tailored to a user's current task, than that text might be better displayed in a printed users' manual.

REFERENCE: Gould and Grischkowsky, 1984; Muter, Latremouille, Treurniet and Beam, 1982.

Consistent Text Format •3

When textual material is formatted, as in structured messages, adopt a consistent format from one display to another.

EXAMPLE: [See sample displays at the end of this section.]

SEE ALSO: 2.0•6.

•4 **▶ Adequate Display Capacity**

When a user must read continuous text on line, display at least four lines of text at one time.

COMMENT: Four lines of text is the minimum which should be displayed when the reading material is simple in content. If the content is more complex, or if a reader will need to refer frequently to previous material, then display more lines of text.

COMMENT: When space for text display is limited, display a few long lines of text rather than many short lines of text.

REFERENCE: Duchnicky and Kolers, 1983.

•5 **▶ Text Displayed in Wide Columns**

Display continuous text in wide columns, containing at least 50 characters per line.

COMMENT: Text displayed in wide columns will be read significantly faster than text displayed in narrow columns.

COMMENT: When space for text display is limited, display a few long lines of text rather than many short lines of text.

REFERENCE: Duchnicky and Kolers, 1983.

SEE ALSO: 2.1•28.

•6 **▶ Conventional Use of Mixed Case**

Display continuous text conventionally in mixed upper and lower case.

EXAMPLE: [See sample displays at the end of this section.]

EXCEPTION: An item intended to attract the user's attention, such as a label or title, might be displayed in upper case.

EXCEPTION: Upper case should be used when lower case letters will have decreased legibility, e.g., on a display terminal that cannot show true descenders for lower case letters.

COMMENT: Reading text is easier when capitalization is used conventionally to start sentences and to indicate proper nouns and acronyms.

REFERENCE: BB 1.6; EG 3.4.3; MS 5.15.3.7.5; Wright, 1977.

•7 **▶ Separation of Paragraphs**

Ensure that displayed paragraphs of text are separated by at least one blank line.

EXAMPLE: [See sample displays at the end of this section.]

REFERENCE: EG 2.3.4; MS 5.15.3.7.3.

▶ Consistent Word Spacing •8

Maintain consistent spacing between the words of displayed text, with left justification of lines and ragged right margins if that is the result.

EXAMPLE: [See sample displays at the end of this section.]

EXCEPTION: Right justification should be employed if it can be achieved by variable spacing, maintaining constant **proportional** differences in spacing between and within words, and consistent spacing between words in a line.

COMMENT: Reading is easier with constant spacing, which outweighs the advantage of an even right margin achieved at the cost of uneven spacing. Uneven spacing is a greater problem with narrow column formats than with wide columns. Uneven spacing handicaps poor readers more than good readers.

REFERENCE: PR 4.5.1, 4.10.5; Campbell, Marchetti and Mewhort, 1981; Gregory and Poulton, 1970; Trollip and Sales, 1986.

SEE ALSO: 1.3•18.

▶ Minimal Hyphenation •9

In display of textual material, keep words intact, with minimal breaking by hyphenation between lines.

COMMENT: Text is more readable if each word is entirely on one line, even if that makes the right margin more ragged.

REFERENCE: BB 3.2; EG 2.2.10.

SEE ALSO: 1.3•19.

▶ Conventional Punctuation •10

Use conventional punctuation in textual display; sentences should end with a period or other special punctuation.

EXAMPLE: [See sample displays at the end of this section.]

REFERENCE: BB 1.3.4; EG 2.2.13.

Clarity of Wording •11

In designing text displays, especially text composed for user guidance, strive for simplicity and clarity of wording.

EXAMPLE: [See sample displays at the end of this section.]

▶ Sentences Begin with Main Topic •12

Put the main topic of each sentence near the beginning of the sentence.

REFERENCE: BB 3.8.2.

2.1 Text

•13

▸ **Simple Sentence Structure**

Use short simple sentences.

EXAMPLE: [See sample displays at the end of this section.]

COMMENT: Long sentences with multiple clauses may confuse the user. Consider breaking long sentences into two or more shorter statements.

REFERENCE: BB 3.8, 3.8.1; EG 2.2.12; Wright, 1977; Wright and Reid, 1973.

SEE ALSO: 4.3•5.

•14

▸ **Concise Wording**

When speed of display output for textual material is slower than the user's normal reading speed, make an extra effort to word the text concisely.

COMMENT: Assume a normal average reading speed of 250 words per minute.

COMMENT: The goal here is to make wording concise but not cryptic. Omitting articles ("the", "a"), prepositions ("of", "by") and relative pronouns ("that", "which", "who") may save some space, but may also reduce comprehension.

REFERENCE: EG 3.3.7.

SEE ALSO: 4.3•5.

•15

▸ **Distinct Wording**

Use distinct words rather than contractions or combined forms, especially in phrases involving negation.

EXAMPLE: (Good) "will not", "not complete"
(Bad) "won't", "incomplete"

COMMENT: This practice will help users understand the sense of a message.

REFERENCE: BB 3.1.4; EG 2.2.15.

•16

▸ **Affirmative Sentences**

Use affirmative statements rather than negative statements.

EXAMPLE:

(Good)

```
Clear the screen before entering data.
```

(Bad)

```
Do not enter data before clearing the screen.
```

COMMENT: Tell the user what to do rather than what to avoid.

REFERENCE: BB 3.8.3; Wright, 1977.

SEE ALSO: 4.0•20.

▶ Active Voice •17

Compose sentences in the active rather than passive voice.

EXAMPLE:
(Good)
Clear the screen by pressing RESET.
(Bad)
The screen is cleared by pressing RESET.

COMMENT: Sentences in the active voice will generally be easier to understand.

REFERENCE: BB 3.8.5; Wright, 1977.

SEE ALSO: 4.0•21

▶ Temporal Sequence •18

When a sentence describes a sequence of events, phrase it with a corresponding word order.

EXAMPLE:
(Good)
Enter LOGON before running programs.
(Bad)
Before running programs enter LOGON.

COMMENT: Temporal order is clearer. Reverse order may confuse a user.

REFERENCE: BB 3.8.6; Wright, 1977.

SEE ALSO: 4.0•22.

Lists for Related Items •19

For a series of related items (words, phrases, instructions, etc.), display those items in a list rather than as continuous text.

COMMENT: A list format will facilitate rapid, accurate scanning.

REFERENCE: BB 1.3.2; Wright, 1977.

•20 ▸ **Single-Column List Format**

Format lists so that each item starts on a new line; i.e., a list should be displayed as a single column.

EXAMPLE:

(Good)
```
Major USI functional areas include:
Data Entry
Data Display
Sequence Control
User Guidance
Data Transmission
Data Protection
```

(Bad)
```
Major USI functional areas include:
Data Entry            Data Display
Sequence Control      User Guidance
Data Transmission     Data Protection
```

EXCEPTION: Listing in multiple columns may be considered where shortage of display space dictates a compact format.

EXCEPTION: Multiple columns of data should be used where that facilitates comparison of corresponding data sets, as in tabular displays (Section 2.3).

REFERENCE: BB 1.9.2; EG 2.3.5; MS 5.15.3.5.6.1.

SEE ALSO: 3.1.3•3.

•21 ▸ **Marking Multiline Items in a List**

When a single item in a list continues for more than one line, mark items in some way so that the continuation of an item is obvious, i.e., so that a continued portion does not appear to be a separate item.

EXAMPLE: Items might be separated by a blank space, or continuing lines within an item might be indented, or each item might be numbered or marked by a special symbol such as an arrow or bullet.

COMMENT: Some demarcation is particularly needed when a list is comprised of a mixture of long and short items.

▶ Arabic Numerals for Numbered Items •22

When listed items will be numbered, use Arabic rather than Roman numerals.

COMMENT: Arabic numbers are more familiar to most users, and therefore require less interpretation than Roman numerals do. The advantage of Arabic numbers becomes greater when large numbers are used. For instance, contrast XXVIII with 28.

REFERENCE: Wright, 1977.

▶ Logical List Ordering •23

Adopt some logical principle by which to order lists; where no other principle applies, order lists alphabetically.

COMMENT: It is the user's logic which should prevail rather than the designer's logic, where those are different.

REFERENCE: EG 2.3.1; MS 5.15.3.5.6.

SEE ALSO: 2.3•2, 2.5•14 thru •18.

▶ Vertical Ordering in Multiple Columns •24

If a list is displayed in multiple columns, order the items vertically within each column.

EXAMPLE:

(Good)	S.R. Abbott	B.M. Drake
	C.N. Abernethy	S.M. Dray
	C.A. Adams	M.G. Dumoff
	H.L. Ammerman	R.C. Eakins
	C.J. Arbak	S.L. Ehrenreich
	etc.	
(Bad)	S.R. Abbott	C.N. Abernethy
	C.A. Adams	H.L. Ammerman
	C.J. Arbak	A.J. Aretz
	A.F. Aucella	J.A. Ballas
	M.C. Bardales	S.H. Barry
	etc.	

▶ Hierarchic Structure for Long Lists •25

For a long list, extending more than one displayed page, consider adopting a hierarchic structure to permit its logical partitioning into related shorter lists.

•26 **Abbreviations Defined in Text**

When words in text displays are abbreviated, define each abbreviation in parentheses following its first appearance.

COMMENT: This practice will help only those users who read displayed text from front to back and remember what they have read. For forgetful users, and for users who sample later sections of a multipage text display, abbreviations may still seem undefined. For such users, it might be helpful to provide an on-line dictionary of abbreviations for convenient reference.

REFERENCE: BB 3.1.8.

SEE ALSO: 4.4•20.

•27 **Highlighting Text**

When a critical passage merits emphasis to set it apart from other text, highlight that passage by bolding/brightening or color coding or by some auxiliary annotation, rather than by capitalization.

COMMENT: A single word might be capitalized for emphasis, but capitalizing an extended passage will reduce its readability.

SEE ALSO: 2.1•6.

•28 **Combining Text with Other Data**

When text is combined with graphics or other data in a single display, thus limiting the space available for text, format the text in a few wide lines rather than in narrow columns of many short lines.

EXAMPLE:

(Good)

```
Text is easier to read when displayed in wide
lines than when displayed in thin columns.
```

(Bad)

```
Text is harder
to read when
displayed in
thin columns
than when
displayed in
wide lines.
```

EXCEPTION: Listed items might be displayed in a narrow column format.

REFERENCE: Duchnicky and Kolers, 1983.

SEE ALSO: 2.1•5.

▶ Placing Figures Near Their Citations •29

When tables and/or graphics are combined with text, place each figure near its first citation in the text, preferably in the same display frame.

EXCEPTION: If a figure is cited at several points in the text, then it might be desirable to allow optional display of the figure at user request, perhaps as a temporary window overlay at each point of citation.

EXCEPTION: If a figure is cited at several points in printed text, and particularly if that text may be accessed at different places by its readers (as in the case of printed reference material), then it might be desirable to group figures consistently at a particular location, such as at the end of each section.

COMMENT: Readers may not bother to find and look at a figure is it is displayed separately from its citation in the text.

REFERENCE: Whalley and Fleming, 1975.

2.1 Text

(Good) **Sample Text Display** **(Good)**

SYSTEM BROADCAST MESSAGES

27 March 1984

Word Processing Users:

Please do NOT print multiple copies of large documents (more than 100 printed pages) during normal working hours. Large print requests will delay printing service for all users.

If you do need bulk printing, submit your request before leaving at 4:30 pm. Your printouts will be ready by the next morning.

Network Users:

Please report any net-related problems to the User Support Center, x2222.

* Press CONTINUE to display the Office Systems Menu.

These sample displays represent a broadcast message received by all users logging onto an on-line office support system. The wording of the bad display is taken from an actual instance. The good display clarifies wording of the text and improves display formatting.

The bad display is annotated to indicate violations of several of the design guidelines proposed here for text display. Although most of its noted deficiencies are minor, in sum they create a display that is potentially confusing to its users.

(Bad) **Sample Text Display** **(Bad)**

```
— System Broadcast Messages —
  SYSTEM #22 - WPS                        27 March 1984

                **** NOTICE ****

      WPS USERS ARE REMINDED  NOT  TO PRINT
  MULTIPLE COPIES OF LARGE SIZE  DOCUMENTS
  (100 PAGES OR MORE) TO THE 6670 PRINTER  OR
  LINEPRINTER  SINCE IT CAUSES  LONG  DELAYS
  ON THOSE PRINTERS.
      IF YOU NEED  MULTIPLE  COPIES,  PLEASE
  SUBMIT  YOUR REQUEST  BEFORE LEAVING  AT
  4:30 P.M.  THANK YOU.
                    ******

  NETWORK USERS — Please  report any  network
  related problems to the  User  Support  Center,
  X2222.

  Questions or problems  should  be  referred to the
  USC, X2222.

  Press the RETURN key to enter the Office Systems
  Menu
█
```

This bad text display violates in some degree several design guidelines in this section:

2.1•1 Conventional text display
•3 Consistent text format
•6 Conventional use of mixed case
•7 Separation of paragraphs
•8 Consistent word spacing
•10 Conventional punctuation
•11 Clarity of wording
•13 Simple sentence structure

2.2 Data Forms

Data forms can display sets of related data items in labeled fields formatted to aid data entry and review.

•1 **Forms for Related Data**

Use forms to display related sets of data items in separately labeled fields.

COMMENT: Forms can aid review of related data items by displaying explanatory labels to caption each data field.

•2 **Visually Distinctive Data Fields**

Provide clear visual definition of data fields, so that the data are distinct from labels and other display features.

EXAMPLE: [See sample displays at the end of this section.]

REFERENCE: MS 5.15.4.3.4.

SEE ALSO: 1.0•6, 1.4•10.

•3 **▶ Data Field Labeling**

Identify each data field with a displayed label.

COMMENT: Do **not** assume that the user can identify individual data fields because of past familiarity. Context may play a significant role: 617-271-7768 might be recognized as a telephone number if seen in a telephone directory, but might not be recognized as such in an unlabeled display.

REFERENCE: BB 1.8.7; EG 2.2.16; MS 5.15.3.1.9.

SEE ALSO: 1.4•5 thru •8, 4.0•11.

•4 **▶ Descriptive Wording of Labels**

Choose a word or phrase to label each field that will describe the data content of that field.

EXAMPLE: [See sample displays at the end of this section.]

COMMENT: Labels should be worded carefully so that they assist users in scanning the display and assimilating information quickly.

REFERENCE: BB 3.5; EG 3.2; MS 5.15.3.1.10.

▶ Consistent Wording of Labels •5

Ensure that labels are worded consistently, so that the same data item is given the same label if it appears in different displayed forms.

EXAMPLE: [See sample displays at the end of this section.]

COMMENT: It may also help to employ consistent grammatical format for different labels; i.e., do **not** use single words or phrases for some labels and short sentences for others, or use verbs for some and nouns for others.

REFERENCE: BB 3.8.4; MS 5.15.3.1.6.

SEE ALSO: 4.0•23.

▶ Distinctive Wording of Labels •6

Ensure that field labels are worded distinctively from one another.

REFERENCE: BB 3.5; EG 3.2.3.

▶ Consistent Label Location •7

Place each label in a consistent location above or to the left of its associated data field(s).

EXAMPLE: In a numbered list, vertically formatted, the numeric labels should be aligned so that the data items start in a fixed column position on the display.

COMMENT: Consistent alignment of labels and data will aid display scanning by a user.

REFERENCE: EG 2.3.9; MS 5.15.3.1.10.a.

SEE ALSO: Section 2.3.

▶ Distinctive Label Format •8

Ensure that labels are distinctive in format/positioning to help users distinguish them from data and other display features.

EXAMPLE: Labels might be capitalized when data are displayed in mixed case, or might be dim when data are bright, or might perhaps be displayed in a different font where that capability exists.

EXAMPLE: [See sample displays at the end of this section.]

REFERENCE: EG 3.2.3; MS 5.15.3.1.10.c, 5.15.4.3.5.

SEE ALSO: 4.0•8.

•9 ▸ **Labels Close to Data Fields**

Ensure that each label is sufficiently close to bc associated with its data field, but is separated from its data field by at least one space.

REFERENCE: BB 1.9.5; EG 2.3.8.

SEE ALSO: 1.4•8.

•10 ▸ **Labeling Units of Measurement**

Include the units of measurement for displayed data either in the label or as part of each data item.

EXAMPLE: `Distance (km): ___ ___ ___`
or
`Distance: ___ ___ ___ km`

REFERENCE: BB 1.8.8; MS 5.15.4.3.10.

SEE ALSO: 1.4•21.

•11 **Consistent Format Across Displays**

Ensure that the ordering and layout of corresponding data fields is consistent from one display to another.

REFERENCE: BB 1.8.4, 2.8.3; MS 5.15.3.1.6.

SEE ALSO: 2.3•12.

•12 ▸ **Form Compatible for Data Entry and Display**

When forms are used for data entry as well as for data display, ensure that the format for data display is compatible with whatever format is used for data entry; use the same item labels and ordering for both.

SEE ALSO: 1.4•24.

Consistent Format Within Data Fields •13

Ensure that the internal format of frequently used data fields is consistent from one display to another.

EXAMPLE: Telephone numbers should be consistently punctuated, perhaps as 213-394-1811.

EXAMPLE: Time records might be consistently punctuated with colons, as HH:MM:SS or HH:MM or MM:SS.S, whatever is appropriate.

EXAMPLE: Date records might be consistently formatted with slashes, as MM/DD/YY.

COMMENT: The convention chosen should be familiar to the prospective users. For European users, the customary format for telephone numbers and dates is different than suggested in the examples above. For military users, date-time data are frequently combined in an accepted special format. For many user groups, time records are kept on a 24-hour clock, which should be acknowledged in display formatting.

REFERENCE: EG 2.2.17.

Partitioning Long Data Items •14

Divide long data items of mixed alphanumeric characters into subgroups of three or four characters separated by a blank or by some special symbol.

EXAMPLE: [See sample displays at the end of this section.]

EXCEPTION: Words should be displayed intact, whatever their length.

COMMENT: Hyphens may be used instead of blanks where that is customary. Slashes are less preferred for separating groups, since they are more easily confused with alphanumerics.

COMMENT: Where a common usage has been established, as in the NNN-NN-NNNN of US social security numbers, grouping should follow that usage.

REFERENCE: EG 2.2.2; MS 5.15.3.1.7, 5.15.3.5.8.

SEE ALSO: 1.0•16.

Distinguishing Blanks from Nulls •15

Distinguish blanks (keyed spaces) from nulls (no entry at all) in the display of data forms, where that can aid task performance.

EXAMPLE: [See sample displays at the end of this section.]

COMMENT: Some special symbol might be adopted to denote null entry. If field delimiters are displayed to guide data entry, then it will often be sufficient simply to leave those delimiters unchanged when no entry has been made.

SEE ALSO: 1.4•10.

2.2 Data Forms

(Good) **Sample Data Form Display** **(Good)**

VISA APPLICATION

NAME: **Jones, Andrew David** VISA: **356 478**
LAST, FIRST MIDDLE

BIRTH COUNTRY: **UK** DATE: **3/22/25**
M D Y

NATIONALITY: **UK** PASSPORT: **Z196284**

ADDRESS: **5 Fairview Lane**
Loughborough, LE11 3RG
England

OTHER TRAVELERS ON THIS VISA

NAME:	BIRTH COUNTRY:	DATE:
Jones, Sandra Jean	**UK**	**10/11/28**
Jones, Cynthia Leigh	**FR**	**6/12/68**
______________________	__	__/__/__
______________________	__	__/__/__
LAST, FIRST MIDDLE		M D Y

* Review and ENTER changes if needed.

These sample displays represent a possible form for review and possible revision of visa application data. In the good display, data entries are bolded to help distinguish them from labels and field delimiters. Fields are ordered consistently in relation to a (supposed) paper application form, and formatted to facilitate data review.

The bad display is annotated to indicate violations of several of the design guidelines proposed here for data form displays. The data entries in the bad display were invented to suggest what a user might have entered, if confused by inadequate labeling and the absence of field delimiters.

(Bad) **Sample Data Form Display** **(Bad)**

```
Name Andrew D. Jones                    Visa Number 356478

Birthplace London                       Nationality English

Passport Z196284                        Birthdate Mar. 22,

Address  1925
         5 Fairview Lane, Loughborough, L
         E11 3RG, England

Other travelers on this visa
Traveler's Name                         Date of Birth - Place
Sandra J. Jones                            Oct. 11, - 1928
Birmingham
Cynthia L. Jones                           June 12, - 1968
Paris, France





Review and ENTER changes if needed
```

This bad data form display violates in some degree several design guidelines in this section:

2.2•2 Visually distinctive data fields
•4 Descriptive wording of labels
•5 Consistent wording of labels
•8 Distinctive label format
•14 Partitioning long data items
•15 Distinguishing blanks from nulls

This bad data form also violates various design guidelines pertaining to data entry, as noted at the end of Section 1.4.

2.3 Tables

Tables can display data in row-column format to aid detailed comparison of ordered sets of items.

•1 **Tables for Data Comparison**

When information handling requires detailed comparison of ordered sets of data, adopt a tabular format for data display.

•2 **Logical Organization**

Organize tabular data in some recognizable order to facilitate scanning and assimilation.

EXAMPLE: Dates might be ordered chronologically, names alphabetically.

EXAMPLE: [See sample displays at the end of this section.]

REFERENCE: BB 1.8.1; EG 2.2.3, 2.3.1; MS 5.15.3.1.4.

SEE ALSO: 2.1•23, 3.1.3•21.

Table Access by Row and Column •3

Construct a table so that row and column labels represent the information a user has prior to consulting the table, i.e., the information that can be used to access table entries for a particular task.

EXAMPLE: If a user's task were to determine characteristics of various raw materials, then a table might be organized as

Raw Material	Transport Costs	Processing Time	Consumer Acceptance
A	High	Slow	Good
B	High	Fast	Good
C	Low	Slow	Good
D	High	Slow	Poor
E	High	Fast	Poor
F	Low	Fast	Poor

whereas if the user's task were to identify what raw material meets certain criteria, then the table might be organized as

		Consumer Acceptance	
		Good	Poor
High Transport Costs	Fast Processing	B	E
	Slow Processing	A	D
Low Transport Costs	Fast Processing		F
	Slow Processing	C	

REFERENCE: Wright, 1977.

▶ Tables Referenced by First Column •4

When tables are used for reference, display the reference items, i.e., those by which the table will be accessed, in the left column; display the material most relevant for user response in the next adjacent column; and display associated but less significant material in columns further to the right.

EXAMPLE: [See sample displays at the end of this section.]

COMMENT: As a corollary, when a list of people is ordered alphabetically by their last name, then their last names should be displayed first, i.e., as the leftmost element in each row.

REFERENCE: Hamill, 1980; Wright, 1977.

•5 ▶ **Items Paired for Direct Comparison**

If data items must be compared on a character-by-character basis, display one directly above the other.

COMMENT: But remember that users will not be entirely accurate in making such comparisons; automated comparison by computer analysis would be more reliable.

REFERENCE: MS 5.15.3.1.8.

•6 **Row and Column Labels**

Label the rows and columns of tabular displays following the guidelines proposed for labeling the fields of data forms.

EXAMPLE: [See sample displays at the end of this section.]

REFERENCE: BB 1.8.7.

SEE ALSO: Section 2.2.

•7 ▶ **Consistent Label Format**

Adopt a consistent format for labeling the rows and columns of displayed tables.

EXAMPLE: Each column label might be left-justified with the leftmost character of the column entries beneath it.

COMMENT: Consistent left justification of column labels will prove especially helpful when columns vary in width.

REFERENCE: Hartley, Young and Burnhill, 1975.

SEE ALSO: 2.0•6, 4.0•6.

•8 ▶ **Distinctive Labeling**

Ensure that row and column labels are distinguishable from the data displayed within tables, and from the labels of displayed lists such as menu options or instructions to users.

COMMENT: There are many ways to distinguish different types of labeled material, including consistent differences in display format/placement as well as special fonts and markers.

REFERENCE: EG 2.2.7.

SEE ALSO: 3.1.3•20.

•9 ▶ **Numbered Items Start with "1"**

When rows or columns are labeled by number, start the numbering with "1", rather than "0".

COMMENT: In counting, people start with "one"; in measuring, they start with "zero".

REFERENCE: EG 2.2.6.

▶ Repeated Elements in Hierarchic Numbering •10

For hierarchic lists with compound numbers, display the complete numbers; do **not** omit repeated elements.

EXAMPLE: (Good)

```
2.1 Position Designation
2.1.1 arbitrary positions
2.1.1.1 discrete
2.1.1.2 continuous
2.1.2 predefined positions
2.1.2.1 HOME
2.1.2.2 other
```

(Bad)

```
2.1. Position Designation
     1. arbitrary positions
        1 discrete
        2 continuous
     2. predefined positions
        1 HOME
        2 other
```

COMMENT: Implicit numbering, as in the "bad" example, may be acceptable for tasks involving perception of list structure. Complete numbering is better, however, for tasks requiring search and identification of individual items in the list.

REFERENCE: Smith and Aucella, 1983b.

▶ Labeling Units of Measurement •11

In tabular displays, either consistently include the units of displayed data in the column labels, or else place them after the first row entry.

EXAMPLE:

(Good)

Time (s)	Velocity (m/s)	Temperature (°C)
5	8	25
21	49	29
43	87	35

(Also acceptable)

Time	Velocity	Temperature
5 s	8 m/s	25 °C
21	49	29
43	87	35

REFERENCE: BB 1.8.8.

•12 Consistent Column Spacing

Maintain consistent column spacing within a table, and from one table to another.

EXAMPLE: [See sample displays at the end of this section.]

EXCEPTION: When columns are grouped under superheadings, it may help to add extra space between superheadings, in order to emphasize that the columns under any single superheading are related.

REFERENCE: BB 1.8.3.

SEE ALSO: 2.2•11.

•13 ▶ Column Scanning Cues

Separate the columns in a table by enough blank spaces, or by some other distinctive feature, to ensure separation of entries within a row.

EXAMPLE: [See sample displays at the end of this section.]

COMMENT: For most tables, a column separation of at least three spaces should be maintained. Certainly the spacing between columns should be greater than any internal spaces that might be displayed within a tabulated data item.

REFERENCE: EG 2.3.6.

•14 ▶ Row Scanning Cues

In dense tables with many rows, insert a blank line (or some other distinctive feature to aid horizontal scanning) after a group of rows at regular intervals.

EXAMPLE: For many applications it will suffice to insert a blank line after every five rows.

EXAMPLE: [See sample displays at the end of this section.]

SEE ALSO: 1.5•10.

Justification of Alphabetic Data •15

Show columns of alphabetic data with left justification to permit rapid scanning.

EXAMPLE: (Good)

```
APL
COBOL
FORTRAN
PL1
```

(Bad)

```
    APL
  COBOL
FORTRAN
    PL1
```

EXCEPTION: Indentation can be used to indicate subordinate elements in hierarchic lists.

EXCEPTION: A short list, of just four or five items could be displayed horizontally on a single line, in the interests of compact display format, if that is done consistently.

REFERENCE: BB 1.3.1; EG 2.2.8, 2.2.11; MS 5.15.3.5.3.

Justification of Numeric Data •16

Justify columns of numeric data with respect to a fixed decimal point; if there is no decimal point, then numbers should be right-justified.

EXAMPLE: [See sample displays at the end of this section.]

REFERENCE: BB 1.4.2, 1.4.3; EG 2.3.9; MS 5.15.3.5.3; PR 4.8.10, 4.10.6.

SEE ALSO: 1.5•7.

▶ Maintaining Significant Zeros •17

When a user must enter numeric values that will later be displayed, maintain all significant zeros; zeros should not be arbitrarily removed after a decimal point if they affect the meaning of the number in terms of significant digits.

REFERENCE: BB 1.4.3.

SEE ALSO: 1.5•8.

2.3 Tables

(Good) **Sample Tabular Display** **(Good)**

AUTOMOBILE OWNERS Page 1 of 4

LICENSE	EMPLOYEE	EXT	DEPT
MA 127 355	Michaels, Allison	7715	91
MA 135 449	Duvet, William	3898	81
MA 227 379	Smithson, Jill	2491	63
MA 227 GBH	Zadrowski, Susan	2687	53
MA 253 198	Jenskin, Erik	3687	31
MA 286 PAM	Hilsmith, Joseph	2443	100
MA 291 302	Leonard, John	7410	92
MA 297 210	Toth, Kelley	7538	45
MA 328 647	Cooksey, Roger	2144	64
MA 342 NCG	Hesen, Christopher	7544	21
MA 367 923	Maddox, Patrick	7070	66
MA 375 NRC	Ashley, Maria	3397	34
MA 376 386	Wheetley, Katherine	2638	58
MA 385 248	Malone, Frank	2144	64
MA 391 293	Lowman, Edward	8263	77

n = Next page

These sample displays represent a table for finding the owner of a car with a particular license plate. (Perhaps it is an employee who has parked in the wrong place, or who has left headlights burning.) In the good display, data entries are ordered by license number to aid the search.

The bad display is ordered alphabetically by employees' last name, which will not prove helpful for this task. The bad display is annotated to indicate several other violations of the design guidelines proposed here for tabular displays.

(Bad) **Sample Tabular Display** **(Bad)**

Automobile Owners			
Sara Alwine	2438	MA 929 448	103
Christopher Aranyi	2716	MA 797 AND	97
Maria Ashley	3397	MA 375 NRC	34
Arlene Atchison	7672	NH 60731	28
Steven Bahr	3272	MA 635 203	35
David Baldwin	3295	NH 63577	75
David Benkley	3581	MA 589 ADE	58
Marlin Boudreau	3413	MA 816 HER	93
Roger Cooksey	2144	MA 328 647	64
Joseph Curran	3167	RI 4693	83
Kent Delacy	3619	MA 749 827	29
Susan Doucette	2797	MA 525 115	41
Joseph Drury	7604	NH 42265	27
William Duvet	3898	MA 135 449	81
Samuel Everett	3619	MA 635 ASK	29
Jeanne Fiske	7642	MA 614 CSU	10
Nancy Graham	2358	MA 745 CKJ	10
Paul Greenbaum	3979	MA 846 BLN	103
Christopher Hesen	7544	MA 342 NCG	21
Joseph Hilsmith	2443	MA 286 PAM	100

This bad tabular display violates in some degree several design guidelines in this section:

2.3•2 Logical organization
•4 Tables referenced by first column
•6 Row and column labels
•12 Consistent column spacing
•13 Column scanning cues
•14 Row scanning cues
•16 Justification of numeric data

Various other guidelines are also violated in this bad table, including those pertaining to identification of multipage displays and display of control options.

2.4 Graphics

Graphics show spatial, temporal, or other relations among data by special formatting of displayed elements.

•1 **Graphic Displays**

Consider graphics rather than text description or tabulation, for display of data showing relations in space or time.

COMMENT: People cannot readily assimilate detailed textual or tabular data, although sometimes such data are necessary. Therefore, a graphic display might be designed where graphic elements showing trends and differences are combined with text annotation and tabular presentation of detailed data. In some applications, it might prove helpful to supplement a primary graphic display with alternative displays of detailed data available as a user-selected option.

REFERENCE: MS 5.15.3.6.1; Foley and Van Dam, 1982; Stewart, 1980.

•2 **▶ Data Comparison**

When users must quickly scan and compare related sets of data, consider graphic format to display the data.

EXAMPLE: Graphic display might help users discern errors in a data base, since deviant "outliers" will appear visually distinct from the body of correct data.

REFERENCE: EG 2.2.9; MS 5.15.3.6.1; Cleveland, 1985.

•3 **▶ Monitoring Data Change**

When users must monitor changing data, consider graphic format to display the data.

COMMENT: Whenever possible, the computer should handle data monitoring and should call abnormalities to the user's attention. When that is not possible, and a user must monitor data changes, graphic display will make it easier for the user to detect critical changes and/or values outside the normal range.

COMMENT: The current lore of graphic design derives chiefly from static, printed displays. Computer processing, however, offers a potential for continuous dynamic display of changing data that should be considered in all methods of graphic presentation.

REFERENCE: Hanson, Payne, Shiveley and Kantowitz, 1981; Tullis, 1981.

Consistency •4

Use consistent logic in the design of graphic displays, and maintain standard format, labeling, etc., for each method of graphic presentation.

COMMENT: Consistency in graphic design will allow users to focus on changes in displayed data without being distracted by changes in display format.

COMMENT: The standardization advocated here has to do with the logic of user interface design, not with internal processing by graphics software. Recent efforts to establish international standards for graphics software have been focused on the internal encoding, processing, storage and transmission of graphics displays in digital form. Such data processing standards do not in themselves specify or significantly constrain user interface design.

REFERENCE: Tufte, 1983.

SEE ALSO: 2.0•6.

Only Necessary Information Displayed •5

Tailor graphic displays to user needs and provide only those data necessary for user tasks.

COMMENT: Current advances in the technology (and theory) of graphic display permit realistic depiction of complex natural scenes. Such technology has been successfully applied to generate displays for arts and entertainment, and may also find increasing application to information displays. For many information displays, however, less may be more: an abstracted schematic diagram, omitting much detail, may convey more effective information than a photographic image. For any particular application, the amount of detail needed should be determined based on a task analysis.

REFERENCE: Tucker, 1984; Tufte, 1983.

SEE ALSO: 2.0•2.

Highlighting Critical Data •6

When a user's attention must be directed to a portion of a graphic display showing critical or abnormal data, highlight that feature with some distinctive means of data coding.

EXAMPLE: On a bar chart, one bar representing an out-of-tolerance condition might be textured or shaded differently to call attention to it and to contrast it with other bars.

COMMENT: More specific recommendations for highlighting different kinds of graphic displays are provided elsewhere in this section.

SEE ALSO: 2.6•1.

•7 **Reference Index or Baseline**

When a user must compare graphic data to some significant level or critical value, include a reference index or baseline in the display.

EXAMPLE: A horizontal line might be displayed on a profit-and-loss graph to indicate where displayed curves exceed the break-even point.

COMMENT: Most data plots include a displayed baseline of some sort. An additional reference index may be displayed as well. The baseline should be chosen with care to provide an appropriate reference for displayed data. A graph without a baseline, or with a poorly chosen baseline, may distort the interpretation of displayed data.

COMMENT: More specific recommendations for indexing different kinds of graphic displays are provided elsewhere in this section.

REFERENCE: Tufte, 1983.

•8 **Text Annotation**

When a graph contains some outstanding or discrepant feature that merits attention by a user, consider displaying supplementary text to emphasize that feature.

EXAMPLE: A flow diagram for process control might include a current advisory message, POSSIBLE PRESSURE VALVE FAILURE, as well as appropriate graphic indications of the problem.

COMMENT: This recommendation derives from the lore of audiovisual aids, where speakers are exhorted to "get the message across" with words as well as pictures. In some information system applications, a graphic display may convey many messages at once. It might then prove difficult to determine which message(s) should be stated in words. As in other aspects of display design, some priorities must be established in relation to the user's information requirements.

REFERENCE: Tufte, 1983.

•9 ▶ **Data Annotation**

When precise reading of a graphic display is required, annotate the display with actual data values to supplement their graphic representation.

EXAMPLE: Adjacent numeric annotation might be added to the ends of displayed bars on a bar graph; numeric data might be displayed to mark the points of a plotted curve.

COMMENT: Some displays may require complete data annotation, but many displays will require annotation only for selected data elements.

▶ Consistent Annotation Format •10

Format any displayed annotation consistently in relation to the graphic elements.

EXAMPLE: Labels might always be placed over the displayed points with which they are associated.

COMMENT: Sometimes it might be necessary to displace a label from its "standard" position to avoid overlap or crowding on the display; such exceptions should themselves be handled consistently.

SEE ALSO: 2.4•4.

▶ Normal Orientation for Labels •11

Display the annotation of graphic displays, including labels for the axes of graphs, in a normal orientation for reading text.

EXAMPLE: For users reading English, labels should be displayed horizontally, even for the vertical axis of a graph.

COMMENT: A conventional text orientation of labels will permit faster, more accurate reading. With a printed graph, it may be possible to tilt the page to read a disoriented label. With an electronic display, a user usually cannot tilt the display screen but instead must tilt his/her head.

COMMENT: More specific recommendations for labeling different kinds of graphic displays are provided elsewhere in this section.

REFERENCE: Noyes, 1980; Tufte, 1983.

Standard Symbols •12

Establish standard meanings for graphic symbols and use them consistently within a system and among systems with the same users.

EXAMPLE: As a negative example, if an aircraft symbol is used to denote an aircraft on one display, that symbol should **not** be used to mark airfields on another display.

COMMENT: If users may be unfamiliar with the graphic symbology used, consider incorporating a legend to define displayed symbols. Alternatively, users might be allowed to request a supplementary display of symbol definitions or to request the definition of a particular displayed symbol by pointing at it.

REFERENCE: Foley and Van Dam, 1982; Hopkin and Taylor, 1979.

2.4 Graphics

•13 ▶ **Pictorial Symbols**

Design pictorial symbols (e.g., icons, pictograms) to look like the objects or processes they represent, and test the resulting symbol set with a representative group of users to ensure that the intended meanings will be understood.

COMMENT: Some pictorial symbols have conventional meanings within a user population, which must be followed to ensure their correct interpretation. Novel symbol design must always be tested. It can happen that a symbol whose meaning seems perfectly clear to its designer may **not** be understood by system users.

REFERENCE: Barnard and Marcel, 1984; Smith, Irby, Kimball and Verplank, 1982.

SEE ALSO: 3.1.8•3.

•14 **Simple Texture Codes**

In selecting textures to code displayed areas, choose simple hatching rather than elaborate patterns.

COMMENT: Compared with manual drafting methods, it is temptingly easy to have a computer generate texture codes of considerable complexity. A designer should resist that temptation. When in doubt, create some sample displays and check them to ensure that texture codes do not produce distracting visual effects such as moire patterns.

COMMENT: Texture coding is a technique specifically related to graphics. Other kinds of display coding — size, shape, brightness, color, etc. — can be applied more generally in display design. Display coding is considered in Section 2.6 of these guidelines.

REFERENCE: Tufte, 1983.

Zooming for Display Expansion •15

When a user may need to perceive graphic relations more accurately, or to view pictures, diagrams, maps, etc. in greater detail, provide a zooming capability that allows the user to expand the display of any selected area.

COMMENT: Zooming can increase display spacing among crowded data items so that they can be perceived better. Thus an air traffic controller might expand a portion of a situation display to see more clearly the spacing of converging tracks that threaten a collision.

COMMENT: Zooming can increase the degree of detail, i.e., can add data to a display. Thus a user might expand a city map to see detailed road structures that are not shown in a small-scale map. When used this way, a zooming capability implies that graphic data be "layered" hierarchically at different levels of aggregation, which may require complex data files and data management techniques.

COMMENT: Zooming might be implemented as a continuous function, by which a display can be expanded to any degree, analogous to a continuous panning capability. Or zooming might be implemented in discrete increments, as in increasing the magnification of an optical instrument to x2, x4, etc. Incremental zooming, with abrupt changes in display scale, may tend to disorient a user, but might prove acceptable in some applications.

SEE ALSO: 2.7.2•13.

▸ Show Changing Scale •16

When a map or other graphic display has been expanded from its normal coverage, provide some scale indicator of the expansion factor.

EXAMPLE: A linear indicator of current map scale might be shown in the margin, or perhaps simply a numeric indication of the display expansion factor (e.g., x 4).

COMMENT: In many applications it may be helpful to show the scale even for a display with normal, unexpanded coverage.

SEE ALSO: 2.7.2•14.

•17 ▶ **Show Overview Position of Visible Section**

When a display has been expanded from its normal coverage, provide some graphic indicator of the position in the overall display of the visible section.

EXAMPLE: In a corner of any frame of an expanded display, the computer might show a rectangle representing the overall display, in which a smaller rectangle is placed to indicate the position and extent of the currently visible portion of that display.

COMMENT: A graphic indication of the current coverage of an expanded display will provide some visual context to help a user maintain a conceptual orientation between the visible part and the whole display from which that part has been expanded.

REFERENCE: Foley and Van Dam, 1982.

SEE ALSO: 2.4.8•11, 2.7.2•15.

•18 **Animation for Dynamic Display**

Consider animation, the movement of data elements under computer control, for displaying a temporal sequence of changing events, or for the pictorial display of complex objects.

EXAMPLE: For air traffic control, sequential frames of radar data might be displayed (with time compression) to aid perception of the tracks from moving aircraft.

EXAMPLE: A complex molecular structure might be perceived more effectively if a viewer is shown sequential displays depicting a computer-stored model from different angles.

EXAMPLE: An architect might demonstrate a proposed building design with a sequential "walk through" displayed from a computer-stored model.

COMMENT: Animation can be used to enhance a variety of graphic displays, including scatterplots, curves, bar graphs, flow charts, etc. Computer tools to support display animation are growing more powerful, and should find increased use in information displays. Prototype testing may be required to determine optimal timing for sequential display, which will vary with different applications.

REFERENCE: MS 5.15.3.6.3; Dunn, 1973; Tucker, 1984.

SEE ALSO: 2.7.3•4.

▶ Highlighting by Animation •19

When sequential relations or other connectivity between display elements requires highlighting, consider animation for that purpose.

EXAMPLE: Connectivity might be emphasized by an arrow moving repeatedly between two displayed elements.

EXAMPLE: Sequential relations might be emphasized by an animated "sprite", i.e., a moving pointer under computer control.

COMMENT: There was a time when viewers of "sing-along" motion pictures were exhorted to "follow the bouncing ball" which marked their place. A moving marker of that kind is now often called a "sprite", or sometimes a "movable object block" (MOB). Sprites can simplify the process of animating computer-generated displays. Once a graphic element has been defined to a computer as a sprite, that element can be moved about a display independently of a fixed background or of other sprites.

COMMENT: If only one element is shown moving on an otherwise stable display, that moving element will be seen as distinctive. Such animated highlighting is probably subject to diminishing returns. If one sprite is good for directing a user's attention, two may **not** be. The simultaneous display of multiple sprites may confuse a user.

Printing Graphic Displays •20

When on-line graphic displays must be printed, allow users to display the material exactly as it will appear in the printed output.

COMMENT: On-line displays can offer some advantages over printed graphics, in terms of animation and highlighting. When a user is preparing a display for printed output, however, it is important that limitations of the print medium can be taken realistically into account. If the printed version does not appear satisfactory, it may be necessary to reformat the display in some way. Alternatively, it may be possible to find a printer with greater capabilities.

2.4.1 Graphics - Scaling

Scaling refers to the positioning of displayed data elements with respect to a defined measurement standard.

•1 **Scaling Conventions**

Follow conventional scaling practice, so that values on an axis increase as they move away from the origin, and the horizontal X-axis is used to plot time or the postulated cause of an event (the independent variable) and the vertical Y-axis is used to plot a caused effect (the dependent variable).

EXAMPLE: In a graph showing plant growth, the X-axis might show successive days, or it might show increasing amounts of water or fertilizer applied.

COMMENT: Scaling conventions also apply to the placement of the origin of a graph. When graphed data represent positive numbers, which is usually the case, the graph should be displayed with the origin at the lower left. When the data include negative values and the axes must extend in both directions from a zero point, that origin should be displayed in the center of the graph.

COMMENT: When the X-axis represents time intervals, the labeled scale points should represent the **end** of each time interval. This consistent usage will aid interpretation of all data plots, including scatterplots, line graphs, and bar charts.

•2 **▶ Consistent Scaling**

If users must compare graphic data across a series of charts, use the same scale for each chart.

COMMENT: Users will find it difficult to compare data sets that are scaled differently. Moreover, users may overlook differences in labeling, and assume that the same scale has been used even when displayed scales are actually different from one another.

COMMENT: Note that in many applications it may prove more effective to display data for comparison in a single combined chart, rather than requiring users to compare data across a series of charts.

REFERENCE: Cleveland, 1985.

Labeling Axes •3

Label each scale axis clearly with its description and measurement units, if any.

EXAMPLE: Labels might include "Population in Thousands", "Price in $1000", "Percent", "Fiscal Year", etc.

COMMENT: Labels should be displayed in conventional text orientation on both the X- and Y-axis for ease of reading.

REFERENCE: MS 5.15.3.6.4; Tufte, 1983.

SEE ALSO: 2.4•11.

Linear Scaling •4

Employ a linear scale for displayed data, in preference to logarithmic or other non-linear methods of scaling.

EXCEPTION: A logarithmic scale shows percentage change rather than arithmetic change; it may be appropriate for some special applications.

COMMENT: Most users are more familiar with linear scales and will interpret linear scales more accurately than other methods of scaling.

▶ Scaling in Standard Intervals •5

Construct scales with tick marks at a standard interval of 1, 2, 5, or 10 (or their multiples by 10) for labeled divisions; intervening tick marks to aid visual interpolation should be consistent with the labeled scale interval.

EXAMPLE: As a negative example, it is **not** acceptable to let the computer divide available scale space automatically if that results in a scale labeled in unfamiliar intervals such as 6 or 13.

EXCEPTION: In special instances, the X-axis might be scaled in odd intervals to show customary divisions, such as the seven days in a week or the 12 months in a year.

COMMENT: Users will find it difficult to interpret scales based on odd intervals, even if computers do not.

▶ Numeric Scales Start at Zero •6

When users must compare aggregate quantities within a display, or within a series of displays, scaling of numeric data should begin with zero.

COMMENT: If for any reason the zero point is omitted, the display should include a clear indication of that omission.

2.4.1 Graphics - Scaling

•7 **Restricted Use of Broken Axes**

When data comparisons of interest fall within a limited range, consider contructing the scaled axis to emphasize that range, with a break in the displayed axis to indicate discontinuity with the scale origin.

COMMENT: Note, however, that a broken axis distorts the displayed amount in relation to a base value and so risks confusing users. In effect, a user will expect that a scale marked in regular intervals will continue in a consistent fashion. If an axis must be broken, label that break clearly, perhaps with some indicator that extends across the displayed graph.

REFERENCE: Cleveland, 1985; Tufte, 1983.

•8 **Duplicate Axes**

When scaled data will contain extreme values, display duplicate axes, so that the X-axis appears at both the top and bottom, and the Y-axis at both the left and right sides of the graph.

COMMENT: Extreme data values may be located far from conventionally placed axes. When duplicate axes are displayed at the top and right side, users will find it easier to read the extreme values.

REFERENCE: Cleveland, 1985; Wright, 1977.

•9 **Single Scale Only**

Design graphs so that only a single scale is shown on each axis, rather than including different scales for different curves in the graph.

EXCEPTION: For users experienced in data analysis, multiple-scale charts may prove an effective tool for comparing relative values of different variables.

COMMENT: Single-scale graphs will generally permit more accurate reading than graphs displaying several scales. Many users will be confused by multiple-scale graphs and make errors when interpreting them. Moreover, by changing the relative scale factors of multiple-scale graphs it is possible to change radically their apparent meaning and bias interpretation by users.

COMMENT: If multiple-scale graphs **are** used, an interactive display capability might aid interpretation, e.g., permitting a user to select any curve and have the computer highlight the corresponding scale for that curve.

▶ Scaling Against a Reference Index •10

If different variables on a single graph seem to require different scales, consider scaling them against a common baseline index, rather than showing multiple scales.

EXAMPLE: Rather than showing sales in units and profits in dollars, both might be graphed in terms of percent change from a baseline period.

COMMENT: An indexed chart can permit comparisons among different variables when multiple scales would otherwise be needed. However, care should be taken in selecting an appropriate base period against which to index, in order to ensure that comparisons will not be biased.

COMMENT: Index scaling may also be appropriate for showing the effect of a single variable (such as money) whose units of measurement change in real value with time.

SEE ALSO: 2.4•7.

Aids for Scale Interpolation •11

Where accuracy of reading graphic data is required, provide computer aids for exact interpolation.

EXAMPLE: It might suffice to allow users to request a fine grid as an optional display feature; or it might be better to display vertical and horizontal rules that a user could move to intersect the axes of a chart; or it might prove best simply to let a user point at any data item and have the computer label that item with a readout of its exact value(s).

▶ Unobtrusive Grids •12

When grid lines are displayed, ensure that they do not look like data and do not obscure data elements — curves, bars, plotted points, etc.

COMMENT: Grid lines should be thinner than data curves, and should be invisible behind depicted objects and areas such as the bars on a bar chart. Note in particular that heavy vertical grid lines may conceal the height of plotted peaks.

COMMENT: In this respect, electronic displays offer more flexibility than printed graphs. Grids can be displayed or suppressed by user selection. For reading the value of a particular data point, perhaps no grid is needed at all. A user might simply ask the computer to display the value of any selected point.

REFERENCE: Cleveland, 1985; Tufte, 1983.

2.4.1 Graphics - Scaling

•13 Restricted Use of Three-Dimensional Scaling

Consider three-dimensional scaling, where a Z-axis is added to the display, only in special applications for experienced users.

COMMENT: Showing a Z-axis on a display that is limited to two actual dimensions will confuse many users. If three-dimensional scaling is employed, adopt a consistent method of representation, e.g., isometric or orthographic projection, perspective drawing, or triangular coordinate grid.

COMMENT: It is often possible in graphic display to show a third dimension through use of auxiliary coding — e.g., color or shape coding, or supplementary annotation — which may prove more effective than trying to represent a third spatial dimension pictorially.

Scatterplots show relations among the individual data points in a two- dimensional array.

Scatterplots •1

Consider scatterplots, in which data are plotted as points in a two-dimensional graph, to display how two variables are correlated or to show the distribution of points in space.

EXAMPLE: A changing display of points representing radar data, such as those used for monitoring aircraft tracks, might be regarded as a dynamic scatterplot.

COMMENT: Curves can be superimposed on scatterplots to indicate computed data trends, correlations, or other derived statistical measures, thus combining two types of graphic display.

COMMENT: Scatterplots, as the name implies, are sometimes used to show a dispersal intended to indicate **non**-correlation of variables. But scatterplots may not be convincing for that purpose, because users will often perceive or imagine patterns in scattered data points where none actually exist.

COMMENT: Note that scatterplots cannot be shown effectively in most forms of three-dimensional spatial representation because of inherent display ambiguities. (Here the triangular grid might be considered an exception.) A third dimension might be represented by coding the symbols used to plot different data categories. If that is done, however, the visual correlation between any two variables in the scatterplot will be obscured.

Highlighting •2

If some plotted points represent data of particular significance, highlight those points to make them visually distinctive from others.

EXAMPLE: Significant data points might be highlighted by bolding, color, blinking, shape coding, or other means, or might be designated by supplementary display annotation.

SEE ALSO: 2.4•6, 2.6•1.

2.4.2 Graphics - Scatterplots

•3 **Grouping Scatterplots to Show Multiple Relations**

When relations among several variables must be examined, consider displaying an ordered group (matrix) of scatterplots, each showing the relation between just two variables.

COMMENT: The ordering of several scatterplots in a single display might help a user discern relations among interacting variables.

REFERENCE: Cleveland, 1985.

•4 ▸ **Interactive Analysis of Grouped Scatterplots**

When scatterplots are grouped in a single display to show relations among several variables, provide some interactive aid for analysis so that if a user selects a set of data in one plot then the corresponding data points in other plots will be highlighted.

COMMENT: Data selection might be accomplished by "brushing" a scatterplot with a superimposed box of controllable size to define the data set of interest. That technique can exploit the capabilities of interactive graphics to permit a range of data analysis not possible when using printed graphs.

REFERENCE: Cleveland, 1985.

Curves and line graphs show relations among sets of data defined by two continuous variables.

Curves and Line Graphs •1

Consider curves (in which data relations are summarized by a smoothed line) or line graphs (in which plotted data points are connected by straight line segments) for displaying relations between two continuous variables, particularly for showing data changes over time.

COMMENT: Line graphs are regarded here merely as a special form of plotted curves, hence recommendations for displaying curves are intended to apply also to line graphs.

COMMENT: Curves are generally superior to other graphic methods for speed and accuracy in interpreting data trends. Unlike printed graphs, computer-generated curves can show dynamic data change, as in oscilloscope displays.

COMMENT: A curve implies a continuous function. Where that could be misleading, a better choice might be a bar graph composed of discrete display elements from one data point to the next.

COMMENT: Sometimes curves may be combined with other graph types. For example, annual sales for the past several years might be displayed as a bar chart, followed by curves to indicate monthly sales during the current year.

REFERENCE: Schutz, 1961.

Comparing Curves •2

When several curves must each be compared with the others, display them in one combined graph.

COMMENT: The objective here is an integrated display that will provide a user with all needed information. On the other hand, as more curves are added to a graph the user's task of comparison will become more difficult. Some designers recommend that no more than four curves be displayed in a single graph. Certainly it is clear that some reasonable limit should be adopted.

COMMENT: If one particular curve must be compared with each of several others, some designers recommend multiple charts in which each pairing is shown separately, rather than displaying all the curves in one single chart. When multiple charts are used for such a purpose, the same scale should be used in each of the charts.

REFERENCE: MS 5.15.3.6.5.

SEE ALSO: 2.4.1•2.

2.4.3 Graphics - Curves and Line Graphs

•3

Labeling Curves

When multiple curves are included in a single graph, each curve should be identified directly by an adjacent label, rather than by a separate legend.

EXCEPTION: Where displayed curves are too close for direct labeling, an acceptable alternative might be to distinguish the various curves in some way, perhaps by color coding or line coding, and identify their codes in a separate legend.

COMMENT: Direct labeling will permit users to assimilate information more rapidly than displaying a separate legend.

REFERENCE: Milroy and Poulton, 1978.

SEE ALSO: 2.4•11.

•4

▶ Compatible Ordering in Legends

If a legend must be displayed, order the codes in the legend to match the spatial order of their corresponding curves in the graph itself.

EXCEPTION: If legends are shown for a series of related graphs, then adopt some logical order consistently for all of those legends.

•5

Highlighting

In charts displaying multiple curves, if one curve represents data of particular significance, highlight that curve.

EXAMPLE: If one curve represents critical/discrepant data, that curve might be displayed with a noticeably thicker line stroke or in a different color.

COMMENT: If line coding is already used to distinguish among multiple curves, then the means of highlighting any particular curve should be selected so that it will not be confused with coding for visual separation. For example, if displayed curves are distinguished by line codes (solid, dashed, dotted, etc.), then one curve might be highlighted by displaying it in a different color.

SEE ALSO: 2.4•6, 2.6•1.

Line Coding to Distinguish Curves •6

When multiple curves are displayed in a single graph, and particularly if those curves approach and/or intersect one another, provide line coding to distinguish one curve from another.

EXAMPLE: Curves might be distinguished by different colors or different line types; commonly recommended line types include solid, dashed, dotted, and dot-dashed.

EXCEPTION: When one curve must be compared with a set of other curves, which need not themselves be distinguished, then it might help to display all the curves with the same line type and highlight the one curve that is intended to be distinctive.

COMMENT: Lines might also be coded by stroke width, including at least wide (bold) and narrow (light), but it is probably better to reserve that distinction for use in distinguishing data curves (bold) from background display of reference grids (light). In particular, do **not** use bold or heavy dots for coding, but reserve those for plotting data points.

COMMENT: Aside from conventional means of display coding, it may be possible to provide aids that are more interactive. For example, the computer might highlight any particular curve selected by a user.

REFERENCE: Cleveland, 1985.

SEE ALSO: 2.6•17.

▸ Consistent Line Codes •7

When coding by line type in a series of displayed charts, use line codes consistently to represent corresponding data.

▸ Broken Lines for Projected Curves •8

When curves represent planned or projected or extrapolated data, show them consistently as broken (dashed or dotted) lines to distinguish them from solid curves representing actual data.

COMMENT: A consistent convention in this regard will make charts easier to interpret.

Reference Index •9

When curves must be compared with some critical value, include a reference index in the chart to aid that comparison.

COMMENT: In such cases, the index might be displayed as a horizontal or vertical line, or perhaps as a reference curve of some kind.

SEE ALSO: 2.4•7.

2.4.3 Graphics - Curves and Line Graphs

•10 Repeating Display of Cyclic Data

Where curves represent cyclic data, consider extending the graph to repeat uncompleted portions of the displayed cycle.

EXAMPLE: A plot showing train arrival times at different stations should be extended beyond a 24-hour cycle as necessary to show the complete schedules of any trains en route at midnight.

COMMENT: The intent here is to allow users to scan any critical portion of the displayed cycle without having to return visually to the beginning of the plot. How much extension is desirable will depend on the particular application. In short, data that are used together should be displayed together.

REFERENCE: Tufte, 1983.

•11 Direct Display of Differences

Where users must evaluate the difference between two sets of data, plot that difference directly as a curve in its own right, rather than requiring users to compare visually the curves that represent the original data sets.

EXAMPLE: If two curves represent income and expenses, then it might help to plot the difference between those curves to show profit (or loss).

COMMENT: Some designers recommend band charts, where two curves are plotted and the area between them is textured or shaded, for applications where the difference between curves is of interest, but where that difference must be interpreted in terms of the absolute values of the two variables.

REFERENCE: Cleveland, 1985.

•12 Surface Charts

When curves represent all of the portions of a whole, consider using a surface chart in which curves are stacked above one another to display aggregated amounts, and the areas defined below the curves are textured or shaded.

EXAMPLE: Surface charts might be appropriate to display sales volume over time in different market areas, or for different products.

COMMENT: The area texture or shading between curves has an implication of volume that is appropriate for many purposes. However, for curves that do not represent parts of a whole (e.g., a set of price indices), surface charts might have misleading implications and should **not** be used.

COMMENT: Surface charts permit smooth, continuous display of data categories that might be represented in more discrete form by a set of segmented bars. Thus, recommendations for surface charts may be applied also to segmented bar charts.

SEE ALSO: 2.4.4•9.

▶ Ordering Data in Surface Charts •13

Order the data categories in a surface chart so that the least variable curves are displayed at the bottom and the most variable at the top.

COMMENT: In a surface chart, any irregularity in the bottom curve will "propagate" throughout the curves above it, which will make it difficult for a user to distinguish whether apparent irregularity in upper curves is real or merely a consequence of this method of presentation.

COMMENT: Sometimes there are independent logical grounds for the ordering of data categories. If a surface chart constructed on a logical basis produces confusing irregularity of curves, then it might be better to display the data in some other graphic format.

SEE ALSO: 2.4.4•10.

▶ Labeling Surface Charts •14

Where space permits, label the different areas of surface charts directly within the textured or shaded bands.

Cumulative Curves •15

Consider cumulative curves to show the current total at any point; but do **not** rely on cumulative curves to show effectively the amount of change at any point.

COMMENT: Cumulative curves tend to "wash out" local variations in the displayed data. The rate of change in incremental data can be estimated by judging the slope of a cumulative curve at any point, but that is hard to do.

2.4.4 Graphics - Bar Graphs

Bar graphs show a comparative measure for separate entities or for a variable sampled at discrete intervals.

•1 **Bar Graphs**

Consider bar graphs, where numeric quantities are represented by the linear extent of parallel bars, for comparing a single measure across a set of several entities or for a variable sampled at discrete intervals.

COMMENT: Displayed bars are usually shown extending from a common origin. For some applications, however, the bars might extend between separately plotted high- and low-points. Bars might be displayed, for example, to indicate the range of observed measures.

COMMENT: The displayed linear extent of adjacent bars permits direct visual comparison of quantity, and thus effective assimilation of comparative data by users.

COMMENT: The value of the bar graph format, as with other graphic displays, is to speed information assimilation by a user. In some applications, however, a user can scan displays in a leisurely way, as when reviewing printed output. In such cases, the data shown in a bar graph could often be presented more economically (i.e., more compactly) by a textual description or in a small table.

COMMENT: For experienced users, the overall pattern of a bar graph may serve a diagnostic function beyond the comparison of individual bars. For example, if multiple bars show data from different components of a complex system, then users may learn characteristic "profiles" of the bars which indicate system status.

•2 **▶ Histograms (Step Charts)**

Consider histograms (step charts), bar graphs without spaces between the bars, when there are a great many entities or intervals to be plotted.

COMMENT: Histograms are often used to plot frequency data, i.e., the frequency of observations for each of many intervals scaled along the X-axis. For such applications, a histogram will avoid the "picket fence" appearance which might result from spaces between bars.

Consistent Orientation of Bars •3

In a related series of bar graphs, adopt a consistent orientation for bars displaying similar information, either vertical or horizontal.

EXAMPLE: Vertical bars can be used to display frequency counts, size, cost, or a large variety of other measured attributes.

EXAMPLE: If bar length is used to represent time duration, then it might be more appropriate to orient the bars horizontally, in accord with the general convention of plotting time on the horizontal axis of a graph.

COMMENT: Consistent orientation will aid users in assimilating information from a series of bar graphs.

COMMENT: Here the phrase "bar graph" is used to denote graphic displays in which bars extend either horizontally or vertically. Some designers distinguish between these two alternatives, calling displays with vertical bars "column charts".

Bar Spacing •4

Space adjacent bars closely enough that a direct visual comparison can be made without eye movement.

COMMENT: In this regard, some designers recommend that the spacing between bars be less than the bar width.

COMMENT: If there are a great many bars to be displayed, then spacing will produce an alternating pattern of bright and dark bands that could prove visually disturbing, particularly for viewers with epileptic vulnerability. In such a case it might be better to display a histogram with no spacing between bars.

Reference Index •5

When the extent of displayed bars must be compared with some critical value, include a reference index in the chart to aid that comparison.

EXAMPLE: A horizontal line might be an adequate reference index for a vertical bar graph.

EXAMPLE: If bars are shown to monitor the pulse rates of patients under intensive care, then **two** reference lines might be displayed to define an acceptable zone.

COMMENT: Indexing may be complicated in situations where the displayed bars do not represent a common measure. In such a case, it might help to choose (or devise) an index scheme so that bar lengths will fall in the same zone under normal conditions, so that deviations in bar length will be readily noticed by users who must monitor changing data.

SEE ALSO: 2.4•7.

2.4.4 Graphics - Bar Graphs

•6 **Highlighting**

In a simple bar graph, if one bar represents data of particular significance, highlight that bar.

EXAMPLE: If one bar represents critical/discrepant data, that bar might be displayed with different tonal coding, such as solid black rather than cross-hatched (or vice versa).

EXCEPTION: If bar coding is already used for other purposes, such as to distinguish among different sets of grouped bars, then no additional highlighting code should be superimposed on the bars themselves; perhaps some other means of highlighting (e.g., an arrow) might be adopted.

SEE ALSO: 2.4•6, 2.6•1.

•7 **Paired or Overlapped Bars**

When paired measures from two data sets must be compared, consider displaying each pair as contiguous or (partially) overlapped bars.

EXAMPLE: A common application of paired data is the display of planned versus actual quantities.

COMMENT: Paired bars will permit a direct visual comparison by the user. When more than two data sets must be compared, a display of grouped bars will be less effective. As the number of matched items becomes larger, it might be better to display the data sets in separate bar graphs, or to allow users to select different sets of data for simultaneous display.

COMMENT: In some applications, a good alternative might be to display directly the difference between paired measures. That is to say, a pair of bars showing income and expenses might be converted to a single bar showing the net difference: a "profit" bar might be displayed extending above a "break-even" index line, and a "loss" might be displayed descending below that line.

COMMENT: In a dynamic display where bar length may change while being displayed, it will probably not be a good design choice to overlap the bars.

SEE ALSO: 2.4.3•11.

▶ Labeling Paired Bars •8

When bars are displayed in pairs, label the bars in one pair directly to distinguish the two entities being compared, rather than displaying a separate legend.

COMMENT: Direct labeling of bars will permit efficient information assimilation by a user. If the user has to refer to a separately displayed legend, interpretation of the chart will be slower and more subject to error.

COMMENT: It will probably be sufficient to label just one pair of bars rather than all of them. Labels should have a conventional orientation for reading text. In a dynamic display where bar length may change while being displayed, label position may have to change accordingly.

SEE ALSO: 2.4•11.

Stacked or Segmented Bars •9

Consider stacked bars, in which differently coded segments are shown cumulatively within a bar, when both the total measures and the portions represented by the segments are of interest.

▶ Ordering Data in Stacked Bars •10

In stacked bars, order the data categories within each bar in the same sequence, with the least variable categories displayed at the bottom and the most variable at the top.

COMMENT: In effect, a series of stacked bars is analogous to the stacked curves of a surface chart, and the same design considerations should apply.

COMMENT: Some designers recommend displaying the **largest** values as the bottom segment. Whatever logic is chosen should be maintained consistently from one display to another.

SEE ALSO: 2.4.3•13.

Restricted Use of Icons •11

Consider using iconic symbols of varying size (rather than simple bars) to represent quantitative values in bar graphs only in special cases when unambiguous icons can be provided and when no interpolation will be necessary.

COMMENT: In general, use of icons to represent quantitative information, such as when a silhouette of a person represents 1000 people in a graph, should be avoided. Icons are often ambiguous, and so must be explained somewhere on the figure. In addition, users will find it difficult to interpolate using icons. If a silhouetted person represents 1000 people, then how many people are represented by a silhouette showing just two legs?

REFERENCE: Wright, 1977.

2.4.5 Graphics - Pie Charts

Pie charts show apportionment of a total into its component parts.

•1 **Restricted Use of Pie Charts**

Consider a pie chart only in special cases to show the relative distribution of data among categories, i.e., for displaying data that represent proportional parts of a whole; but note that a bar graph will permit more accurate interpretation for such applications.

COMMENT: There are several significant limitations to a pie chart — in itself it provides no means of absolute measurement, it cannot represent totals greater than 100 percent, and it can only represent a fixed point in time.

COMMENT: Estimation of angular relations, as required in pie charts, is less accurate than estimation of linear extent. Pie charts may have artistic merit in some applications, but will not support accurate assimilation of data.

COMMENT: If pie charts **are** used, some designers recommend that partitioning be limited to five segments or less.

COMMENT: Multiple pie charts will **not** permit accurate comparison of different totals, although different-sized pies can be used to indicate gross differences. Stacked bar graphs will prove more effective for this purpose and should be used when it is necessary to show proportions of different totals.

REFERENCE: Cleveland, 1985.

SEE ALSO: 2.4.4•9.

•2 **Labeling Pie Charts**

If pie charts **are** used, label the segments directly rather than by a separate legend, in a normal orientation for reading text.

SEE ALSO: 2.4•11.

•3 **▶ Numeric Labels**

If pie charts **are** used, add numbers to their segment labels to indicate the percentage and/or absolute values represented in the display.

COMMENT: Because pie charts include no explicit reference scale or index, the only way they can convey numeric values accurately is through their labeling.

Highlighting •4

If a particular segment of a pie chart requires emphasis, highlight it by special hatching or shading and/or by "exploding" it, i.e., displacing it slightly from the remainder of the pie.

SEE ALSO: 2.4•6, 2.6•1.

2.4.6 Graphics - Pictures and Diagrams

Pictures and diagrams show relations among the various elements of objects or processes.

•1 **Pictures**

Use pictorial displays in applications where it is necessary to show accurately detailed representations of real or imaginary objects or processes.

EXAMPLE: Pictorial displays aid in the analysis of objects and events, as in the case of photo interpretation.

EXAMPLE: Pictorial displays support a variety of computer-aided design applications, including the design of aircraft, artificial hearts, automobiles, etc.

COMMENT: In some applications it may be possible for a computer to synthesize pictorial displays from stored data. This is true in computer-aided design, where the computer must display a designed object that does not yet exist. For a map-reading task, a computer might synthesize perspective views of terrain from topographic data, where real images are not available.

COMMENT: For some information-handling tasks the display of detailed images (photographs) will help users. In other instances, simplified line drawings may be more readily comprehended.

COMMENT: Dynamic display of "moving pictures" can exploit various animation techniques to improve the pictorial representation of complex objects and processes.

REFERENCE: Foley and Van Dam, 1982.

•2 **Diagrams**

Use diagrams to show spatial relations, with selective focus on the data specifically required by a user's task, in applications where a full pictorial rendering might be unnecessarily complicated.

EXAMPLE: Diagrams are used to support many applications, ranging from mechanical assembly/maintenance instruction to the representation of electronic circuitry, or floor plans, or organizational hierarchies.

COMMENT: Diagrams are here considered a special form of picture. Diagrams should be kept as simple as possible, omitting unnecessary data. A system diagram for a subway engineer would have to be quite complex, showing accurate distances and directions, and perhaps the relation between subway sections and surface streets, utility lines, etc. But a subway diagram intended for a tourist might display simply the connecting links between one station and another.

SEE ALSO: 2.0•2, 2.4•5.

Linking Sectional Diagrams •3

When diagrammed data exceed the capacity of a single display frame and must be shown in separate sections, provide an overview for the diagram, a consistent notation to indicate the logical linking of its various sections, and an easy means for users to move from one section to another.

EXAMPLE: The sections of a diagram might be assigned letter codes, which could be shown in the overview and at any internal branch points, and which could be entered by a user to request the display of various sections.

COMMENT: General guidelines for linking sectional displays by paging/scrolling are considered elsewhere under the topic of display framing. The concern here is to establish a coherent structure to show logical relations among the separately displayed parts of a diagram. The solution may require internal notation and perhaps replication of some portions of the diagram from one section to another.

COMMENT: An alternative approach might be to construct a hierarchic diagram with a zooming capability to show greater detail. That capability represents a potential advantage of computer-generated electronic displays over printed diagrams.

SEE ALSO: 2.4•15, and Section 2.7.2.

Highlighting •4

If a picture or diagram contains data of particular significance, implying a special need for user attention, highlight those data.

EXAMPLE: Selected portions of pictures might be highlighted by adding a box outline to the display, or perhaps a blinking arrow; diagram elements might be highlighted by bolding or video reversal, or perhaps by color coding.

EXAMPLE: Highlighting might be used to indicate a computer analysis of design deficiencies in a depicted object, or an assessment of damage when monitoring a system that has been degraded in some way.

SEE ALSO: 2.4•6, 2.6•1.

2.4.6 Graphics - Pictures and Diagrams

•5 **Rotation**

In an application where a user must examine a depicted object from different viewpoints, allow the user to rotate its displayed image.

COMMENT: The axis of rotation will generally be the center of the depicted object. Where that is not the case, some indication of the rotation axis should be displayed. In some applications it might also help the user to display some explicit separate indication ("gnomon") of the degree of rotation and the current orientation of the depicted object.

COMMENT: In applications where a user must make a detailed comparison of two (or more) displayed objects, it may be necessary to allow independent rotation, translation and superposition of their images.

REFERENCE: Foley and Van Dam, 1982; Foley, Wallace and Chan, 1984.

•6 **Aids for Pictorial Analysis**

When users must analyze pictorial images in detail, provide appropriate computer aids for that purpose.

EXAMPLE: For examining displayed surfaces, it might be helpful to allow a user to specify/control the light source adopted for computer-generated images.

EXAMPLE: For photo interpretation, computer processing can sometimes improve the visual quality of stored images, as by edge enhancement.

EXAMPLE: For examining the component structure of a complex object, as for an assembly or maintenance task, it might be helpful to allow a user to "explode" an aggregate display into its several elements.

EXAMPLE: For examining the internal structure of a depicted object, it might be helpful to allow a user to request auxiliary displays of specified cross-sections or transect diagrams.

EXAMPLE: For more detailed structural analysis of depicted objects, it might be necessary to provide computer aids for calculating area, volume, center of gravity, modes of vibration, stresses, heat transfer, etc.

Flowcharts are diagrams that illustrate sequential relations among elements or events.

Flowcharts •1

Consider flowcharts for schematic representation of sequence information, i.e., to display data that are logically related in terms of sequential processes.

EXAMPLE: Flow charts are frequently used for process control, project scheduling, and other similar applications.

▶ Flowcharts to Aid Problem Solving •2

Consider providing a flowchart to aid problem solving when a solution can be reached by answering a series of questions, and when no tradeoffs will be required.

EXAMPLE: In process control, a flowchart might aid problem diagnosis when a user must determine the cause of abnormal conditions and take appropriate action.

COMMENT: A flowchart can add structure to complex problem solving by illustrating a set of discrete decision points. With such a flowchart, a user is given specific steps to follow in solving a problem, helping to ensure that all relevant factors are considered. For simple problems, however, a tabular or text format may be read more quickly than a flowchart.

COMMENT: Flowcharts are **not** useful when a user must make tradeoffs. For example, if a user must evaluate alternative outcomes if s/he could invest more money or could accept lower quality, then using a flowchart would be cumbersome and time consuming. When a user must evaluate alternatives, a tabular format may be more efficient.

REFERENCE: Wright, 1977.

Logical Ordering of Steps •3

Design flowcharts so that their displayed steps follow some logical order.

EXAMPLE: When a flowchart displays some sequential process, then display the steps in the order of that sequence.

EXAMPLE: When a flowchart is provided as a problem solving aid, then steps might be ordered by importance, where those decisions which will have the largest impact on the final problem solution are made first, or ordered by certainty, where decisions which can be made with the most certainty are made first.

REFERENCE: Wright, 1977.

2.4.7 Graphics - Flowcharts

•4 ▶ **Ordering to Minimize Path Length**

When there is no inherently logical order to the steps in a flowchart, order them to minimize flowchart size, i.e., to minimize average path length.

COMMENT: Flowchart size can sometimes be reduced by placing first those steps that represent binary decisions.

COMMENT: When all decision options are not equally probable, consider minimizing the length of the most likely path, i.e., that most frequently followed, rather than minimizing the overall size of the flowchart.

REFERENCE: Wright, 1977.

•5 ▶ **Conventional Path Orientation**

Design flowcharts so that the path of the logical sequence is consistent with familiar orientation conventions, i.e., from left to right (for users accustomed to reading English) and from top to bottom, or perhaps clockwise.

REFERENCE: Krohn, 1983.

•6 **Consistent Coding of Elements**

When it is necessary to distinguish different types of flowchart elements, adopt a consistent coding scheme for that purpose.

EXAMPLE: Shape coding of "boxes" in a flowchart is commonly used to indicate the different user actions in an operational sequence diagram that displays the results of task analysis.

COMMENT: Flowchart coding within any system should conform to established conventions and user expectations, with some flexibility to meet changing user requirements. For some applications, such as the operational sequence diagrams noted in the example above, flowcharts may have to meet normative standards established across systems.

REFERENCE: Geer, 1981.

•7 ▶ **Conventional Use of Arrows**

In flow charts and other graphics displays, use arrows in a conventional fashion to indicate directional relations in the sequential links between various elements.

Highlighting •8

If one element in a flowchart represents data of particular significance, implying a special need for user attention, highlight that element.

EXAMPLE: Line coding by color or bolding might be used to highlight displayed paths, and/or the boxes or other graphic elements representing displayed states.

EXAMPLE: Highlighting might be used to distinguish scheduled vs. actual accomplishment, emphasizing critical time delays, cost overruns, or other discrepant conditions.

EXAMPLE: As a cautionary example, the flowchart instructions for cleaning an electric lawnmower might highlight a box which says "unplug it before touching the blade".

COMMENT: Color coding may be particularly appropriate in flowcharts, because of the effective primacy of color for guiding the visual scanning required to trace paths.

SEE ALSO: 2.4•6, 2.6•1.

Single Decision at Each Step •9

When a flowchart is designed so that a user must make decisions at various steps, require only a single decision at each step, rather than combining decisions to reduce flowchart size.

COMMENT: Combining decisions in a single step, such as asking

```
Does the temperature exceed 500 °C
and the pressure exceed 2250 psi?
```

may save space. But such a choice might confuse a user who will be uncertain what path to follow if a situation meets only one of the combined criteria, i.e., in this example, if the temperature is above 500 but the pressure is below 2250.

REFERENCE: Wright, 1977.

•10 **Logical Ordering of Options**

When a flowchart is designed so that a user must make decisions at various steps, display the available options in some logical order.

EXAMPLE:

(Good) Temperature at inlet valve (^{0}F):
h = more than 400
a = 300-400
l = less than 300

(Bad) Temperature at inlet valve (^{0}F):
a = 300-400
h = more than 400
l = less than 300

EXAMPLE: If options represent stages of a process, those stages should be listed in the order in which they would actually occur.

COMMENT: The ordering of options should **not** be determined merely by the amount of space that is conveniently available to display them.

•11 ▶ **Consistent Ordering of Options**

When a flowchart is designed so that a user must make decisions at various steps, display the available options in some consistent order from step to step.

EXAMPLE: "Yes" might always be on the left and "no" on the right.

COMMENT: The point here is that for options which have no inherently logical order, some order should be consistently imposed. Consistent ordering will permit a user to review a flowchart more quickly.

•12 ▶ **Consistent Wording**

Choose some consistent format for wording the options displayed at the decision points in a flowchart.

EXAMPLE: Decision options might be worded as questions, such as

Will this car be driven more than
100 miles a week?

or worded as statements listing options, such as

Car will be driven:
h = more than 50 miles per week
a = 25 to 50 miles per week
l = less than 25 miles per week

COMMENT: Sometimes it may not be possible to use a consistent format for displaying options. However, the more consistent a flowchart can be made in format and wording, the easier it will be to understand and use.

Maps and situation displays are a form of diagram showing geographic relations among objects or events.

Maps •1

Provide maps to display geographic data, i.e., direction and distance relations among physical locations.

COMMENT: Here the term "map" refers to the display of relatively stable geographic data, or the display of slowly changing data such as weather. When mapped data change more quickly, as in the display of aircraft tracks for air traffic control, those diagrams are called "situation displays". Design recommendations for maps will generally pertain also to situation displays.

SEE ALSO: 2.4.8•15.

Consistent Orientation •2

When several different maps will be displayed, adopt a consistent orientation so that the top of each map will always represent the same direction.

EXAMPLE: In common use, most maps are oriented so that North is upward.

Consistent Projection •3

When maps represent large geographic areas, adopt a consistent method of projecting the earth's curvature on the flat display surface.

COMMENT: For large areas, any method of projection will introduce some distortion of apparent distances in the display. The projection used should be one which minimizes the practical effects of that distortion for the application at hand. If a selected method of map projection is used consistently, viewers can learn to compensate in some degree for the distortion it introduces.

REFERENCE: Hopkin and Taylor, 1979.

2.4.8 Graphics - Maps and Situation Displays

•4 **Labeling Maps**

Label significant features of a map directly in the display, when that can be done without cluttering the display.

COMMENT: When labeling and other desired annotation are too extensive to incorporate in the map itself, that material might be shown in some separate, peripheral area of the display. In that case, auxiliary coding might be used to link annotation with associated map features, e.g., by matching symbol codes.

COMMENT: An alternative to fixed labeling would be to permit user interaction with computer-generated graphic displays. If a user selected a mapped point, a label might be temporarily displayed. If a user selected a marginal label, its associated map location might be highlighted.

COMMENT: With interactive graphics, it may be possible to design maps with hierarchic levels of portrayed detail and labeling, so that a user can "zoom in" to examine an area in greater detail or "zoom out" for an aggregated overview.

SEE ALSO: 2.4•10, 2.4•11.

•5 ▶ **Consistent Positioning of Labels**

Position labels on a map consistently in relation to the displayed features they designate.

EXAMPLE: The names of cities and towns might always be placed immediately above the corresponding symbols showing their locations.

COMMENT: As a practical matter, map displays can get very crowded. It may not always prove feasible to maintain a consistent placement for labels, with the result that designers will be tempted to put labels wherever they will fit. In such a crowded display, labels may obscure map features, and vice versa. Locating and reading labels will be slowed, particularly when map features are displayed closely adjacent to the beginning of labels. Under these circumstances, some other approach to map labeling should be considered to avoid crowding.

REFERENCE: Noyes, 1980.

Area Coding •6

When different areas of a map must be defined, or when the geographic distribution of a particular variable must be indicated, consider the use of texture patterns or color or tonal codes for that purpose.

EXAMPLE: Coding might help define different areas of interest, such as the Eastern, Central, Mountain and Pacific time zones on a map of the United States, or perhaps areas of current rainfall on a weather map.

EXAMPLE: Area coding might be used to indicate a geographic variable such as elevation or a demographic variable such as population.

COMMENT: In many applications it may be desirable to limit area coding to one variable in order to assure effective information assimilation, in which case a designer should select that variable which is most significant. Another approach might be to allow a user to specify which variable will be coded on a map and to change that selection at will depending upon current task requirements. In some special applications, however, it may be feasible to superimpose several kinds of area coding to permit multivariate data analysis by skilled users.

SEE ALSO: Section 2.6.

▶ Tonal Codes •7

Where users must make relative judgments for different colored areas of a display, consider employing tonal codes (different shades of one color) rather than spectral codes (different colors).

EXAMPLE: For reading topographic maps, tonal variation might be used to show the relative height of displayed areas, perhaps with darker hues used to code greater heights.

COMMENT: This recommendation represents an exception to other guidelines advocating distinctive code values. Coding by tonal variation should be considered only for applications where perception of relative differences along a single dimension is more important than perception of absolute values.

COMMENT: People can order categories along a continuous dimension to match tonal variations in one color, whereas people do not have a natural means of ordering **different** colors.

COMMENT: This recommendation is based on research with layer tints on printed displays. Some testing may be required to determine whether a particular electronic display permits effective variation in color tones.

REFERENCE: Phillips, 1982.

SEE ALSO: 2.6•24.

2.4.8 Graphics - Maps and Situation Displays

•8 ▶ **Ordered Coding**

Where different areas of a map are coded by texture patterns or tonal variation, order the assigned code values so that the darkest and lightest shades correspond to the extreme values of the coded variable.

EXAMPLE: For indicating the height of mapped terrain, dark shades might be used for high elevations and light shades for low.

COMMENT: Orderly assignment of code values will help users perceive and remember the categories represented by the code.

•9 **Highlighting**

If one area in a map represents data of particular significance, implying a special need for user attention, highlight that area.

EXAMPLE: On a weather map, special area coding might be used to indicate severe storm conditions.

SEE ALSO: 2.4•6.

•10 **Panning for Flexible Display Framing**

When a map exceeds the capacity of a single display frame, in terms of the required extent and detail of coverage, consider providing users a capability to pan the display frame over the mapped data in order to examine different areas of current interest.

COMMENT: General guidelines for viewing different parts of an extended display by paging or panning/scrolling are considered elsewhere under the topic of display framing. Panning will permit users to move continuously over a map in any desired direction, without encountering any internal boundaries imposed by predefined display framing.

COMMENT: An alternative approach might be to define various sections of a large map and link those sections by the same techniques used for linking sections of a large diagram. One risk in that approach is that an area of interest might be at a boundary where none of the sectional maps provide adequate coverage. Thus some degree of overlapped coverage is probably needed at the boundaries of displayed map sections.

SEE ALSO: 2.4.6•3, 2.7.2•12.

▶ Show Overview Position of Visible Section •11

When a user pans over an extended display in order to view different sections, provide some graphic indicator of the position in the overall display of the visible section.

EXAMPLE: In a corner of a panned display, the computer might show a rectangle representing the overall display, in which a smaller rectangle is placed to indicate the position and extent of the currently visible portion of that display.

COMMENT: While panning across a map, moving from one section to another, a user may lose track of what is being displayed, and be uncertain how to move in order to see some other area of interest. An indicator of current position will help maintain user orientation.

SEE ALSO: 2.4•17, 2.7.2•15.

Aiding Distance Judgments •12

When a user must judge distances accurately on a map or other graphic display, provide computer aids for that judgment.

COMMENT: Some designers display for this purpose a movable grid whose scale is controlled automatically by the computer depending upon current map expansion. Some designers have suggested that the grid might consist of a vertical and a horizontal line displayed across a series of concentric range rings. It might be more helpful to display a scaled line (ruler) that a user could move to intercept any two displayed points directly.

COMMENT: For exact measurement, it might be better to allow a user to select (point at) any two points and have the computer "read-out" their separation distance directly. The same technique might be used to determine the direction (bearing) between two points.

Aids for Analyzing Maps •13

When the use of mapped data may be complex, provide computer aids for data analysis.

EXAMPLE: In topographic analysis, computers might analyze contour data at different sites to calculate slopes and sun exposure for land use planning, or might calculate sight angles to determine radar coverage from different locations, etc.

2.4.8 Graphics - Maps and Situation Displays

•14 **Mapping Nongeographic Data**

In applications where the geographic distribution of nongeographic data must be displayed, consider adding other graphic elements to a map for that purpose.

EXAMPLE: A symbol might be displayed in different sizes to indicate a particular measure in different localities, such as average population age, or average annual rainfall, or incidence of a particular disease.

EXAMPLE: Small stacked bars might be superimposed on the different areas of a map to indicate the local distribution of some data measure, such as population distribution by age, education, or income.

COMMENT: Alphanumeric characters might be added to a map to show data, but those will not aid a direct visual comparison across areas in the same way that graphic symbols can do. Moreover alphanumeric data may be confused with labels and other kinds of annotation.

•15 **Situation Displays**

When it is necessary to show the geographic location of changing events, which is often the case for monitoring real situations, combine event data with a map background.

EXAMPLE: A display for air traffic control might superimpose aircraft tracks on a background of geographic coordinates, with supplementary annotation and/or coding to indicate track identification, speed, heading, altitude, etc.

•16 **Indicating Data Change**

When changes in mapped data are significant for a user's task, include auxiliary graphic elements to indicate those changes.

EXAMPLE: Auxiliary coding might be needed to indicate the movement of storm fronts on a weather map.

COMMENT: In dynamic maps, i.e., situation displays, data changes involving movement can be shown directly. On static displays, arrows can be added to indicate the direction of movement of mapped elements. Where movement over an extended area must be represented, as in showing weather fronts, directional "pips" can be added to displayed contour lines. Some designers recommend that such pips should be displayed one to two times as large as alphanumeric characters, and that pip spacing should be five to ten times the pip width.

SEE ALSO: 2.6•20, 2.7.3•4.

Selectable Data Categories •17

If the particular data categories required at different stages in a user's job cannot be exactly predicted, allow the user to select the categories needed for any information handling task.

EXAMPLE: In monitoring aircraft separation for collision avoidance, a user might choose to display selectively the aircraft tracks within a particular altitude zone.

EXAMPLE: To help identify an unrecognized aircraft track, a user might choose to add flight plan data temporarily to an air traffic display.

COMMENT: This recommendation does not lessen the designer's responsibility for determining user needs. Where user information requirements can be defined, displays should be designed to provide necessary data. User selection from available data categories may represent desirable flexibility to meet changing task requirements. However, there is a risk of "data indigestion" if a user selects the wrong categories or too many categories.

COMMENT: Categories of selectable data might include relatively stable elements (e.g., defined flight paths, zones of radar coverage, etc.) as well as the variable elements that represent changing data (e.g., aircraft tracks).

COMMENT: If auxiliary coding is adopted for different data categories, such as shape coding of symbols, code values should be chosen to be distinctive for any likely **combination** of displayed categories.

COMMENT: Users should be provided a ready reminder of what data categories are available, and an easy means of selecting (or suppressing) displayed categories. This implies category selection by menu or function keys.

SEE ALSO: 2.7.1•5.

Stable Reference for Changing Data •18

When graphic data are changing and displays are automatically updated, provide some stable display elements, e.g., coordinates, geographic boundaries, etc.

EXAMPLE: For vehicular control, a common display convention is to depict a moving element (aircraft) against a fixed background (horizon).

COMMENT: Stable display elements provide a frame of reference to help users assimilate and interpret data changes.

COMMENT: Moving data may overlay and temporarily obscure other display elements, such as fixed background data. When that happens, the display update logic must determine which data categories have priority on the display and which may be obscured by others, and should restore the obscured elements when the overlaid data moves away, and should further ensure that no data are erased from the display in the process of obscuring and restoring data.

SEE ALSO: 2.7.3•1.

2.5 Format

Format refers to the organization of different types of data in a display to aid assimilation of information.

•1 **Consistent Format**

Adopt a consistent organization for the location of various display features from one display to another.

EXAMPLE: One location might be used consistently for a display title, another area might be reserved for data output by the computer, and other areas dedicated to display of control options, instructions, error messages, and user command entry.

EXCEPTION: It might be desirable to change display formats in some distinctive way to help a user distinguish one task or activity from another, but the displays of any particular type should still be formatted consistently among themselves.

COMMENT: The objective is to develop display formats that are consistent with accepted usage and existing user habits. Consistent display formats will help establish and preserve user orientation. There is no fixed display format that is optimum for all data handling applications, since applications will vary in their requirements. However, once a suitable format has been devised, it should be maintained as a pattern to ensure consistent design of other displays.

REFERENCE: BB 1.1, 1.8.5; EG 2.2.5, 2.3, 2.3.3; MS 5.15.3.2.1, 5.15.3.3.4; Foley and Van Dam, 1982; Stewart, 1980.

SEE ALSO: 4.0•6.

•2 **Distinctive Display Elements**

Make the different elements of a display format distinctive from one another.

EXAMPLE: Different display areas ("windows") can be separated by spacing (where space permits); outlining can also be used to separate different areas, so that displayed data, control options, instructions, etc., are distinct from each other.

REFERENCE: EG 2.3; MS 5.15.3.1.5; Stewart, 1980.

SEE ALSO: 3.0•8, 4.0•8.

▶ Spacing to Structure Displays •3

Use blank space to structure a display.

COMMENT: Closely packed data are difficult to locate and difficult to read. Thus blank space can be used to advantage to separate different groups of data. Related data items within a group, however, should be displayed close enough to minimize eye movement time in scanning those data.

REFERENCE: Tullis, 1983.

Paging Crowded Displays •4

When a display contains too much data for presentation in a single frame, partition the data into separately displayable pages.

COMMENT: And provide convenient control procedures to allow users to move easily from one page to another.

REFERENCE: BB 4.4.1, 4.4.2; Stewart, 1980.

SEE ALSO: 2.7.2•6.

▶ Related Data on Same Page •5

When partitioning displays into multiple pages, take into account the type of data, and display functionally related data items together on one page.

COMMENT: This recommendation is easily followed for text displays, involving primarily the elimination of "widows", considerate placement of illustrations, etc. For displayed data forms and tables, data grouping and continuation of headers must be considered. The partitioning of graphics displays may require radical redesign.

▶ Page Labeling •6

In a multipage display label each page to show its relation to the others.

SEE ALSO: 2.7.2•5, 2.7.2•6, 4.2•7.

Integrated Display •7

When coherent display is required to aid user perception of relations among data sets, provide those data in an integrated display rather than partitioning them into separate windows.

REFERENCE: EG 2.3.2.

SEE ALSO: 2.7.2•1.

•8 **User-Defined Data Windows**

When the need to view several different kinds of data jointly cannot be determined in advance, allow a user to define and select separate data windows that will share a single display frame.

COMMENT: Depending upon user needs (and system capability), data windows might appear simultaneously as segments of a joint display, might be overlaid in varying degrees so as to obscure one another, or might be displayed sequentially at the user's option. In the latter condition, multiple display windows will differ little from multiple display pages, except perhaps in speed of sequential access.

SEE ALSO: 2.7.5•3.

•9 **▶ Adequate Window Size**

When a display window must be used for data scanning, ensure that the window can display more than one line of data.

REFERENCE: Elkerton, Williges, Pittman and Roach, 1982.

•10 **Display Title at Top**

Begin every display at the top with a title or header, describing briefly the contents or purpose of the display; leave at least one blank line between the title and the body of the display.

REFERENCE: BB 1.1.1, Table 1; PR 4.5.2.

SEE ALSO: 4.2•6.

•11 **Command Entry, Prompts, Messages at Bottom**

Reserve the last several lines at the bottom of every display for status and error messages, prompts, and command entry.

COMMENT: Assuming that the display is mounted physically above the keyboard, which is the customary placement, a user can look back and forth from keyboard to display more easily when prompts and command entry area are at the bottom of the display.

COMMENT: As a corollary to this recommendation, when separate command sets are associated with different display windows, such as options for display control (size of the window, positioning, etc.), those should be shown at the bottom of each window.

REFERENCE: PR 4.5.3; Granda, Teitelbaum and Dunlap, 1982.

SEE ALSO: 3.1.5•2, 4.0•7.

Logical Data Organization •12

Ensure that displays are formatted to group data items on the basis of some logical principle, considering trade-offs derived from task analysis.

REFERENCE: BB 1.8.1.

▶ Grouping for Data Comparison •13

If users must analyze sets of data to discern similarities, differences, trends, and relationships, structure the display format so that the data are consistently grouped.

EXAMPLE:
(Good)

Cost		Output	
Actual	Predicted	Actual	Predicted
947	901	83	82
721	777	57	54
475	471	91	95

(Bad)

Cost		Output	
Actual	Predicted	Predicted	Actual
947	901	82	83
721	777	54	57
475	471	95	91

REFERENCE: BB 1.8.6; Tullis, 1981.

▶ Data Grouped by Sequence of Use •14

Where displayed data are used in some spatial or temporal order, consider grouping those data by sequence of use to preserve that order.

EXAMPLE: Data in an electronic display should match the order of items in an associated paper data form.

REFERENCE: BB 1.8.2; PR 4.10.7.

SEE ALSO: 1.4•25, 1.4•27, 2.4.7•5.

▶ Data Grouped by Function •15

Where sets of data are associated with particular questions or related to particular functions, consider grouping each set together to help illustrate those functional relationships.

REFERENCE: BB 1.8.2; Tullis, 1981.

•16 ▶ **Data Grouped by Importance**

Where some displayed data items are particularly important, i.e., provide significant information and/or require immediate user response, consider grouping those items at the top of the display.

REFERENCE: BB 1.8.2; Tullis, 1981.

•17 ▶ **Data Grouped by Frequency**

Where some data items are used more frequently than others, consider grouping those items at the top of the display.

COMMENT: Principles of data grouping also apply to the display/listing of control options.

COMMENT: These recommendations for data grouping in display formatting follow familiar human engineering principles for display/control layout in equipment design.

REFERENCE: BB 1.8.2.

SEE ALSO: 3.1.3•21.

•18 ▶ **Data Grouped Alphabetically or Chronologically**

When there is no appropriate logic for grouping data by sequence, function, frequency or importance, adopt some other principle such as alphabetical or chronological grouping.

REFERENCE: BB 1.8.1.

SEE ALSO: 2.1•23

Coding refers to distinctive means for highlighting different categories of displayed data for user attention.

Highlighting Critical Data •1

Provide distinctive coding to highlight important display items requiring user attention, particularly when those items are displayed infrequently.

EXAMPLE: Such items might include recently changed data, or discrepant data exceeding acceptable limits, or data failing to meet some other defined criteria.

COMMENT: "Highlight" is used here in its general sense, meaning to emphasize or make prominent, and is **not** restricted to any particular method of display coding such as brightening or inverse video.

COMMENT: Highlighting is most effective when used sparingly, adding emphasis to a display which is relatively uniform in appearance except for just a few highlighted items.

COMMENT: For some purposes position coding, i.e., displaying important items consistently in a particular location, might be a sufficient means of highlighting, as when an error message appears in a space otherwise left blank. But auxiliary codes may still be needed to highlight important items, even if they are positioned consistently.

REFERENCE: EG 2.1.3, 2.3.12; MS 5.15.3.3.1.

▸ Removing Highlighting •2

If highlighting is used to emphasize important display items, remove such highlighting when it no longer has meaning.

EXAMPLE: If highlighting identifies an error, remove that highlighting when the error is corrected.

Coding by Data Category •3

Provide display coding in applications where a user must distinguish rapidly among different categories of displayed data, particularly when those data are distributed in an irregular way on the display.

2.6 Coding

•4 **Meaningful Codes**

Adopt meaningful or familiar codes, rather than arbitrary codes.

EXAMPLE: A three-letter mnemonic code (DIR = directory) is easier to remember than a three-digit numeric code.

COMMENT: An arbitrary code, such as a Social Security Number, may eventually become familiar through frequent use.

REFERENCE: BB 3.6.2; MS 5.15.3.3.1.

•5 ▶ **Familiar Coding Conventions**

Adopt codes for display (and entry) that conform with accepted abbreviations and general user expectations.

EXAMPLE: Use M for "male", F for "female", rather than arbitrary digits 1 and 2. In color coding, use red for danger.

COMMENT: If in doubt, an interface designer can survey prospective users to determine just what their expectations may be.

SEE ALSO: 2.6•32, 4.0•14.

•6 **Definition of Display Codes**

When codes are assigned special meaning in a display, provide a definition at the bottom of the display that replicates the code being defined.

EXAMPLE: The legend on a map is a common example.

EXAMPLE: For a color code each definition should be displayed in its appropriate color, as RED = hostile displayed in red.

REFERENCE: BB 7.6.1.

SEE ALSO: 4.4•21.

•7 **Consistent Coding Across Displays**

Assign consistent meanings to symbols and other codes, from one display to another.

COMMENT: When coding is **not** consistent, the user's task of display interpretation may be made more difficult than if no auxiliary coding were used at all.

REFERENCE: BB 3.6.1, 7.6.2; MS 5.15.3.3.1.

SEE ALSO: 2.0•14, 4.0•13.

Alphanumeric Coding •8

Consider alphanumeric characters for auxiliary coding in display applications such as graphics where the basic data presentation is not already alphanumeric.

COMMENT: Select alphanumeric codes that are visually distinct for visual displays, and phonetically distinct for auditory displays (or in any application where displayed codes must be spoken).

REFERENCE: EG Table 1.

SEE ALSO: 1.0•18.

▶ Consistent Case in Alphabetic Coding •9

For alphabetic codes display all letters consistently either in upper case or in lower case.

COMMENT: For data display, upper case labels may be somewhat more legible. For data entry, computer logic should **not** distinguish between upper and lower case codes, because users find it hard to remember any such distinction.

REFERENCE: BB 1.3.3.

SEE ALSO: 1.0•27.

▶ Combining Letters and Numbers •10

When codes combine both letters and numbers, group letters together and numbers together rather than interspersing letters with numbers.

EXAMPLE: Letter-letter-number ("HW5") will be read and remembered somewhat more accurately than letter-number-letter ("H5W").

COMMENT: Unfortunately, there are common instances in which this practice has not been followed, such as the coding of British and Canadian postal zones.

REFERENCE: BB 1.5.1; MS 5.15.3.5.8.

▶ Short Codes •11

When arbitrary codes must be remembered by the user, ensure that they are no longer than four or five characters.

EXCEPTION: When a code is meaningful, such as a mnemonic abbreviation or a word, it can be longer.

REFERENCE: BB 1.5.2; MS 5.15.3.5.8.

SEE ALSO: 1.0•15.

2.6 Coding

•12 **Special Symbols**

Consider using special symbols, such as asterisks, arrows, etc., to draw attention to selected items in alphanumeric displays.

SEE ALSO: 4.3•19.

•13 ▸ **Consistent Use of Special Symbols**

When using special symbols to signal critical conditions, use them **only** for that purpose.

SEE ALSO: 2.6•7.

•14 ▸ **Markers Close to Words Marked**

When a special symbol is used to mark a word, separate the symbol from the beginning of the word by a space.

COMMENT: A symbol immediately adjacent to the beginning of a word will impair legibility.

REFERENCE: Noyes, 1980.

•15 **Shape Coding**

Consider coding with geometric shapes to help users discriminate different categories of data on graphic displays.

COMMENT: Approximately 15 different shapes can be distinguished readily. If that "alphabet" is too small, it may be possible to use component shapes in combination, as in some military symbol codes.

REFERENCE: EG Table 1.

•16 ▸ **Establishing Standards for Shape Coding**

When shape coding is used, assign codes based on established standards or conventional meanings.

EXAMPLE: A number of international, national, and organizational standards for shape coding exist, and those should be followed where they apply.

COMMENT: Although shape codes can often be mnemonic in form, their interpretation will generally rely on learned association as well as immediate perception. Existing user standards must be taken into account by the display designer.

REFERENCE: MS 5.15.3.3.6.

SEE ALSO: 2.6•7, 4.0•13.

Line Coding •17

For graphic displays, consider using auxiliary methods of line coding, including variation in line type (e.g., solid, dashed, dotted) and line width ("boldness").

COMMENT: Perhaps three or four line types might be readily distinguished, and two or three line widths.

REFERENCE: EG 2.3.

SEE ALSO: 2.4.3•6.

▶ Underlining for Emphasis •18

When a line is added simply to mark or emphasize a displayed item, place it **under** the designated item.

COMMENT: A consistent convention is needed to prevent ambiguity in the coding of vertically arrayed items.

COMMENT: For words composed from the Roman alphabet, underlining probably detracts from legibility less than would "overlining".

REFERENCE: MS 5.15.3.3.5.

▶ Coding by Line Length •19

Consider using codes with lines of varying length for applications involving spatial categorization in a single dimension.

EXAMPLE: The length of a displayed vector might be used to indicate speed.

COMMENT: Perhaps four lengths can be reliably distinguished in practical use. Long lines will add clutter to a display, but may be useful in special applications.

REFERENCE: EG Table 1.

▶ Coding by Line Direction •20

Consider using codes with lines of varying direction for applications involving spatial categorization in two dimensions.

EXAMPLE: The angle of a displayed vector might be used to indicate direction, i.e., heading or bearing.

COMMENT: Users can make fairly accurate estimates of angles for lines displayed at ten-degree intervals.

REFERENCE: Smith, 1962a.

SEE ALSO: Section 1.2.

2.6 Coding

•21 **Limited Use of Size Coding**

Consider size coding, i.e., varying the size of displayed alphanumerics and other symbols, only for applications where displays are not crowded.

COMMENT: Perhaps as many as five sizes might be used for data categorization, but two or three will probably prove the practical limit.

REFERENCE: EG Table 1; MS 5.15.3.3.6.

•22 **▶ Adequate Differences in Size**

For size coding, a larger symbol should be at least 1.5 times the height of the next smaller symbol.

COMMENT: An increase in symbol height must usually be accompanied by a proportional increase in width to preserve a constant aspect ratio and so facilitate symbol recognition.

REFERENCE: MS 5.15.3.3.6.

•23 **Limited Use of Brightness Coding**

Consider coding by differences in brightness for applications that only require discrimination between two categories of displayed items; i.e., treat brightness as a two-valued code, bright and dim.

EXAMPLE: A data form might display dim labels and bright data items, in order to facilitate data scanning.

COMMENT: Perhaps as many as four brightness levels might be used, but at some risk of reduced legibility for the dimmer items. It will be safer to reserve brightness as a two-valued code, particularly for displays whose overall intensity can be adjusted at the terminal by a user.

REFERENCE: EG 2.1.4, Table 1.

SEE ALSO: 1.4•16.

▶ Brightness Inversion •24

When a capability for brightness inversion is available (so-called "reverse video"), where dark characters on a bright background can be changed under computer control to bright on dark, or vice versa, consider brightness inversion for highlighting critical items that require user attention.

COMMENT: Brightness inversion is obviously limited to use as a two-valued code, i.e., a displayed item is either shown with standard or inverted brightness. If brightness inversion is used for alerting purposes, as recommended here, it should be reserved consistently for that purpose, and **not** be used for general highlighting.

REFERENCE: PR 3.3.4.

SEE ALSO: 2.6•7.

Color Coding for Relative Values •25

When the relative rather than the absolute values of a variable are important, consider displaying gradual color changes as a tonal code to show the relative values of a single variable.

EXAMPLE: In displaying ocean depth, a saturated blue might be used to show the deepest point, with gradually desaturated blues to show decreasing depth.

COMMENT: A gradual change in color might be achieved by varying saturation, starting with a saturated hue and gradually adding white. Or the change might be in intensity, starting with an intense hue and gradually adding black. Or the change might be in hue, moving gradually from red through orange, yellow, green, etc.

COMMENT: People can easily make relative color comparisons when colors are displayed simultaneously. However, people find it more difficult to make absolute color judgments. Part of the problem is color **naming**. A particular blue-green hue might be named "green" by one person but "blue" by another. In the example above, a user could not be expected to associate any particular shade of blue with a specific ocean depth.

COMMENT: Gradual color changes should not be used when absolute values are important, or to code data into discrete categories. As an example, in displaying revenues to determine the point at which a company becomes profitable, it would be more appropriate to use two distinctly different colors, such as black and red, with the color changing at the point of profitability.

SEE ALSO: 2.4.8•7.

•26 Color Coding for Data Categories

When a user must distinguish rapidly among several discrete categories of data, particularly when data items are dispersed on a display, consider using a unique color to display the data in each category.

EXAMPLE: Different colors might be used effectively in a situation display to distinguish friendly, unknown, and hostile aircraft tracks, or alternatively to distinguish among aircraft in different altitude zones.

COMMENT: Color is a good auxiliary code, where a multicolor display capability is available. A color code can be overlaid directly on alphanumerics and other symbols without significantly obscuring them. Color coding permits rapid scanning and perception of patterns and relationships among dispersed data items.

COMMENT: Perhaps as many as 11 different colors might be reliably distinguished, or even more for trained observers. As a practical matter, however, it will prove safer to use no more than five different colors for category coding.

COMMENT: With some display equipment now providing millions of different colors, designers may be tempted to exploit that capability by using many different colors for coding. The capability to display many colors may be useful for depicting complex objects, and for providing tonal codes to show the relative values of a single variable. However, such a capability is not useful for coding discrete categories, except that it may allow a designer to select more carefully the particular colors to be used as codes.

REFERENCE: BB 7.2; EG Table 1; MS 5.15.3.3.7; Smith, 1962b; Smith, 1963a; Smith and Thomas, 1964; Smith, Farquhar and Thomas, 1965.

▶ Easily Discriminable Colors •27

When selecting colors for coding discrete categories of data, ensure that those colors are easily discriminable.

EXAMPLE: On a light background, a good choice of colors might be red, dark yellow, green, blue and black; on a dark background, good colors might be a somewhat desaturated red, green and blue, plus yellow and white.

COMMENT: The harder it is for a user to identify a displayed color, the less useful will be the color code. If many colors are used, colors will be closer in hue and harder to discriminate. If color coding is applied to symbols that subtend small visual angles, which makes color perception difficult, there will be a special need to limit the number of colors used.

COMMENT: Varying saturation and intensity in addition to hue may increase the discriminability of colors. For instance, on a dark background a designer might select a saturated yellow but a desaturated red.

COMMENT: Colors selected for coding should be tested with users to ensure that they are easily discriminable. Testing should be conducted under realistic conditions, since such factors as display type and ambient room lighting will affect color perception. If colors will be used for displaying text, care should be taken to ensure that colored letters are legible as well as discriminable.

▶ Conservative Use of Color •28

Employ color coding conservatively, using relatively few colors and only to designate critical categories of displayed data.

COMMENT: Casual, arbitrary use of colors on every display may cause displays to appear "busy" or cluttered. Casual use of color will also reduce the likelihood that significant color coding on particular displays will be interpreted appropriately and quickly by a user.

REFERENCE: BB 7.1.

▶ Adding Color to Formatted Displays •29

Add color coding after displays have already been designed as effectively as possible in a monochrome format.

COMMENT: Do **not** use color coding in an attempt to compensate for poor display format. Redesign the display instead.

REFERENCE: BB 7.3.

2.6 Coding

•30 ▶ **Redundant Color Coding**

Make color coding redundant with some other display feature such as symbology; do **not** code only by color.

COMMENT: Displayed data should provide necessary information even when viewed on a monochromatic display terminal or hard-copy printout, or when viewed by a user with defective color vision.

REFERENCE: BB 7.4, 7.6.3; MS 5.15.3.3.7.

•31 ▶ **Unique Assignment of Color Codes**

When color coding is used, ensure that each color represents only **one** category of displayed data.

COMMENT: Color will prove the dominant coding dimension on a display. If several **different** categories of data are displayed in red, say, they will have an unwanted visual coherence which may hinder proper assimilation of information by a user.

REFERENCE: BB 7.6.1; Smith and Thomas, 1964.

•32 ▶ **Conventional Assignment of Color Codes**

Choose colors for coding based on conventional associations with particular colors.

EXAMPLE: In a display of accounting data, negative numbers might be shown as red, corresponding to past use of red ink for that purpose.

EXAMPLE: Red is associated with danger in our society, and is an appropriate color for alarm conditions. Yellow is associated with caution, and might be used for alerting messages or to denote changed data. Green is associated with normal "go ahead" conditions, and might be used for routine data display. White is a color with neutral association, which might be used for general data display purposes.

COMMENT: Other associations can be learned by a user if color coding is applied consistently.

REFERENCE: BB 7.7.1, 7.7.2, 7.7.3; MS 5.15.4.6.1.f.

SEE ALSO: 2.6•5, 4.0•13, 4.0•14, 4.3•19.

▶ Brightness and Saturation to Draw Attention •33

Use brighter and/or more saturated colors when it is necessary to draw a user's attention to critical data.

COMMENT: On some display equipment, designers can vary the intensity as well as the saturation for individual hues. On those displays, both intensity and saturation can be used to draw a user's attention to critical data.

COMMENT: Although saturated and/or intense hues are useful for drawing a user's attention, their overuse will result in a display which is garish and difficult to view for long periods.

COMMENT: When deciding the desired saturation of any given display color, consider the ambient lighting under which the display will be viewed. Colors that appear highly saturated in a darkened room will appear less saturated when viewed under high ambient illumination.

▶ Saturated Blue for Background Color •34

Use saturated blue only for background features in a display, and **not** for critical data.

EXAMPLE: Saturated blue might be used for shading background areas in graphic displays, where its lower apparent brightness could possibly be of benefit. Or saturated blue might be used to display a background grid or familiar scale on a graphic display.

COMMENT: The human eye is not equally sensitive to all colors, nor are its optics color-corrected. Blue symbols appear dimmer than others, and are more difficult to focus.

COMMENT: If blue must be used for displayed data, use a desaturated blue or cyan in order to make the data more legible.

REFERENCE: BB 7.6, 7.7.5; Weitzman, 1985.

Blink Coding •35

Consider blink coding when a displayed item implies an urgent need for user attention.

COMMENT: If used sparingly, blinking symbols are effective in calling a user's attention to displayed items of unusual significance. Blinking characters may have somewhat reduced legibility, and may cause visual fatigue if used too much.

COMMENT: Perhaps three or four blink rates might be reliably distinguished, but it will probably prove safer to use blinking as a two-level code, blinking versus nonblinking.

REFERENCE: BB 1.10.2, 1.10.3; EG Table 1; MS 5.15.3.3.2; Smith and Goodwin, 1971b; Smith and Goodwin, 1972.

SEE ALSO: 4.3•19.

2.6 Coding

•36 ▶ **Blinking Marker Symbols**

When a user must read a displayed item that is blink coded, consider adding an extra symbol such as an asterisk to mark the item, and then blinking that marker symbol rather than blinking the item itself.

COMMENT: This practice will draw attention to an item without detracting from its legibility.

REFERENCE: BB 1.10.3; Smith and Goodwin, 1971b.

•37 ▶ **Optimal Blink Rate**

When blink coding is used, select a blink rate in the range from 2 to 5 Hz, with a minimum duty cycle (ON interval) of 50 percent.

COMMENT: Although equal ON and OFF intervals are often specified, an effective code can probably be provided even when the OFF interval is considerably shorter than the ON (a "wink" rather than a blink), as in occulting lights used for Navy signaling.

REFERENCE: BB 1.10.4; MS 5.15.3.3.2.

•38 **Coding with Texture, Focus, Motion**

Consider other visual coding dimensions for special display applications, including variation in texture, focus, and motion.

COMMENT: Texture can be useful for area coding in graphic displays. Only two levels of focus are feasible, clear and blurred, with the risk that blurred items will be illegible. Perhaps 2 to 10 degrees of motion might be distinguished, in applications where motion is an appropriate and feasible means of display coding.

REFERENCE: EG 2.3.

SEE ALSO: Section 2.4.

Auditory Coding •39

Consider auditory displays as a means of supplementing visual display, or as an alternative means of data output in applications where visual displays are not feasible.

EXAMPLE: Auditory signals may be helpful in alerting users to critical changes in a visual display.

EXAMPLE: Auditory output might be used to permit telephone access to computer-stored data.

EXCEPTION: Auditory display may be impractical in situations where high ambient noise prevents accurate listening.

COMMENT: As compared with visual displays, an auditory display offers a potential advantage in attracting a user's attention; a user does not have to "listen at" an auditory display in order to hear it. On the other hand, auditory displays suffer from a number of comparative disadvantages. Auditory displays generally do not offer as great a range of coding options. Auditory displays do not permit easy scanning to discern critical data items, or items that may have been missed at first listening. For human listeners with normal vision, auditory displays do not provide a natural representation of spatial relations.

REFERENCE: MS 5.15.3.9.1.

SEE ALSO: 1.3•30, 4.0•26 thru •29.

▶ Distinctive Auditory Coding •40

For auditory displays, employ distinctive sounds to code items requiring special user attention.

EXAMPLE: A variety of signals may be available, including sirens, bells, chimes, buzzers, and tones of different frequency.

COMMENT: Tones may be presented in sequence to enlarge the signal repertoire.

REFERENCE: Smith and Goodwin, 1970.

SEE ALSO: 4.3•19.

▶ Voice Coding •41

For auditory displays with voice output, consider using different voices to distinguish different categories of data.

COMMENT: At least two voices, male and female, could be readily distinguished, and perhaps more depending upon fidelity of auditory output, and listening conditions.

REFERENCE: Simpson, McCauley, Roland, Ruth and Williges, 1985; Smith and Goodwin, 1970.

•42 ▸ **Coding Synthesized Voice Alarms**

If computer-generated speech output is used for auditory display, add a special alerting signal to introduce voice alarms and warning messages in order to distinguish them from routine advisory messages.

REFERENCE: Hakkinen and Williges, 1984; Simpson and Williams, 1980.

SEE ALSO: 4.0•26 thru •29.

Display control refers to procedures by which a user can specify what data are shown, and how.

Flexible Display Control by User •1

When user information requirements cannot be exactly defined by the designer, allow users to tailor displays flexibly on line by controlling data selection, data coverage within a display frame, data updating and suppression.

COMMENT: Here user control of data display is distinguished from the broader control of transaction sequences covered in Section 3 of these guidelines.

COMMENT: Display control by users is certainly a necessary capability in general-purpose data processing systems. In a designed information system, i.e., a system created to perform specific tasks, the need for display control by users will depend on the degree to which users' information requirements can be anticipated by designers. In effect, if you know what data the system users will need, then design the displays to provide those data. If you are uncertain about user requirements, then provide users with some flexibility to control display configuration themselves.

COMMENT: Some more specialized methods of display control (e.g., rotation) are discussed elsewhere in guidelines pertaining to graphic data entry.

SEE ALSO: 2.0•1, 2.0•8, 2.8•1, and Section 1.6.

2.7.1 Display Control - Selection

> *Selection refers to the means for specification of data outputs, either by a user or automatically.*

•1 **User Selection of Data for Display**

For general data processing systems, allow users to specify the data for displayed output; for specific information handling applications, allow users to select data for display as needed to meet task requirements.

COMMENT: Flexibility of data selection by users can be exercised, of course, only within the limits of what data are available within a computer system, and what means for data selection have been provided by the designer.

SEE ALSO: 2.0•2, 2.0•8, 2.8•1.

•2 **Display Identification Labels**

When a user will select data displays, assign to each display an identifying label, i.e., an alphanumeric code or abbreviation that can facilitate display requests by the user.

COMMENT: An identifying label will help users remember different displays and provide a convenient means for requesting them. Even in systems where users exercise little initiative in data selection, where displays are largely configured in advance by designers, some kind of display identification will help users understand the displayed consequences of sequence control actions.

COMMENT: It may also the helpful to include an identifying label in any separately selected "window" that might be overlaid on another display, as noted in Section 2.7.5.

REFERENCE: BB 1.2.3, MS 5.15.3.1.13.

SEE ALSO: 4.2•6.

•3 ▶ **Meaningful Display Labels**

The display identification label should be unique, short, but meaningful enough to be remembered easily.

COMMENT: As conceived here, the display label serves as a shorthand identification. The label does not take the place of a full, descriptive title. The full title would be displayed separately.

COMMENT: Where flexibility is desired, it may be good practice to let a user assign names to the particular sets of data that constitute commonly used displays, either as formal names or else as nicknames associated by the computer with the formal names.

SEE ALSO: 2.5•10, 4.2•6.

▶ Consistent Format for Display Labels •4

Place the identifying label used for display selection in a prominent and consistent location on each display.

EXAMPLE: The top left corner of the display might be used for this purpose.

REFERENCE: BB 1.2.4.

SEE ALSO: 2.5•1, 4.0•6, 4.2•6.

Selectable Data Categories •5

If the particular data categories required at different stages in a user's job cannot be exactly predicted, allow the user to select the categories needed for any information handling task.

EXAMPLE: In monitoring aircraft separation for collision avoidance, a user might choose to display selectively the aircraft tracks within a particular altitude zone.

EXAMPLE: To help identify an unrecognized aircraft track, a user might choose to add flight plan data temporarily to an air traffic display.

COMMENT: This recommendation does not lessen the designer's responsibility for determining user needs. Where user information requirements can be defined, displays should be designed to provide necessary data. User selection from available data categories may represent desirable flexibility to meet changing task requirements. However, there is a risk of "data indigestion" if a user selects the wrong categories or too many categories.

COMMENT: Categories of selectable data might include relatively stable elements (e.g., defined flight paths, zones of radar coverage, etc.) as well as the variable elements that represent changing data (e.g., aircraft tracks).

COMMENT: If auxiliary coding is adopted for different data categories, such as shape coding of symbols, code values should be chosen to be distinctive for any likely **combination** of displayed categories.

COMMENT: Users should be provided a ready reminder of what data categories are available, and an easy means of selecting (or suppressing) displayed categories. This implies category selection by menu or function keys.

SEE ALSO: 2.4.8•17.

2.7.1 Display Control - Selection

•6 **Fast Response to Display Request**

Ensure that system response to simple requests for data display take no more than 0.5 to 1.0 second.

COMMENT: When display output is paced in segments (blocks, paragraphs, etc.), response time refers to output of the first segment. The recommended response time of 0.5 to 1.0 second should apply when a user makes a request (usually perceived as "simple") for the next page of a multipage display, or when a display begins to move in response to a scrolling command. The response to a request for a **new** display may take somewhat longer, perhaps 2 to 10 seconds, particularly if the user perceives such a request to involve more complicated operations, such as accessing different files, transforming data, etc.

COMMENT: The desirability of rapid display generation is often discounted in practice, particularly for general purpose, time-shared systems where other practical design considerations may dictate slower computer response. For dedicated systems, however, those designed to help perform defined information handling tasks, rapid response should be an explicit design goal, with computer output capabilities designed accordingly.

REFERENCE: EG Table 2.

SEE ALSO: 3.0•14, 4.2•2, 4.2•3.

•7 **▶ Signaling Completion of Display Output**

If display generation is slow, notify the user when display output is complete.

EXAMPLE: A nonobtrusive auditory signal such as a chime should suffice for this purpose.

•8 **Regenerating Changed Data**

Where the computer must regenerate a display to update changed data items, consider regenerating only those changed items if that will speed display output.

COMMENT: The critical design issue here is speed of display. It may be easier for the computer to regenerate an entire display than to change just one item. But if that results in slower computer response to the user, then the added work of selective display regeneration may be worth-while.

COMMENT: Partial display regeneration to show data changes should only be used, of course, when that can be accomplished without erasing **un**changed data.

▶ Initial Erasure to Replace Changed Data •9

When the computer must regenerate a display to update changed data, erase old data items before adding new data items to the display.

COMMENT: The aim here is to avoid any momentary user confusion that might result from watching portions of old data being overwritten and partially overlapped by portions of new data.

▶ Nondestructive Overlay •10

If changing data elements temporarily overlay and obscure other display data, ensure that the obscured data are **not** permanently erased but will reappear if the overlaid data are later removed.

EXAMPLE: In a situation display moving track data may temporarily obscure stable background data.

COMMENT: To govern the appearance of data overlay, it may be necessary to establish a hierarchy of data categories to determine which will be shown when displayed in combination. Actively changing data will generally take priority over more stable background/reference data.

SEE ALSO: 2.4.8•15, 2.7.5•10.

Printing Displays Locally •11

When displayed data are of potential long-term interest, give users some easy means to print paper copies locally, within security restraints.

COMMENT: Users should not have to remember displayed data. Optional printout allows a user to record data from one display to compare with another, and so deal with situations where a designer has not anticipated the need for such comparison.

COMMENT: A user should **not** have to take notes or transcribe displayed data manually. That practice underutilizes the data handling potential of the computer, and risks transcription errors by the user.

REFERENCE: BB 4.4.6; EG 4.2.14; MS 5.15.9.2; PR 4.10.1.

SEE ALSO: 6.2•7, 6.4•7.

2.7.2 Display Control - Framing

Framing refers to user control of data coverage by display movement, including paging, scrolling, offset, etc.

•1 **Integrated Display**

In designing displays, include all data relevant to a user's current transaction in one display frame or page.

COMMENT: This recommendation is particularly important when critical data items must be compared by a user. Do **not** rely on a user to remember data accurately from one display frame to another.

COMMENT: If a user requests a display output that exceeds the capacity of a single frame, than it is obviously not possible for any designer to ensure an integrated display. However a designer can mitigate the problems associated with use of extended displays, by providing effective means for identifying and controlling sequential access to different portions of the display.

REFERENCE: EG 3.4.4.

SEE ALSO: 2.0•1, 2.5•5, 2.5•7, 4.0•5, 4.4•1.

•2 **Easy Paging**

When requested data exceed the capacity of a single display frame, give users some easy means to move back and forth over displayed material by paging or panning/scrolling.

EXAMPLE: Dedicated function keys might be provided for paging forward and back.

COMMENT: Note that critical data requiring integrated display for effective assimilation should be included in a single frame, and **not** dispersed over several pages. Paging is acceptable when the user is looking for a specific data item, but **not** when the user must discern some relationship in a set of data.

REFERENCE: BB 4.4.1, 4.4.2, 4.4.9; EG 6.3.8; MS 5.15.3.1.11.

•3 **▸ Continuous Numbering in Multipage Lists**

When a list of numbered items exceeds one display page, number the items continuously in relation to the first item on the first page.

REFERENCE: EG 2.3.10.

▸ Labels for Multipage Tables •4

For a large table that exceeds the capacity of one display frame, ensure that users can see column headings and row labels in all displayed sections of the table.

REFERENCE: BB 1.9.6; MS 5.15.3.5.4.

SEE ALSO: 1.5•1.

Annotating Display of Continued Data •5

When lists or tables are of variable length, and may extend beyond the limits of one display frame, inform a user when data are continued on another page and when data are concluded on the present page.

EXAMPLE: Incomplete lists might be marked

`continued on next page` or

`continued` or

`more`

while concluding lists might display a note

`end of list` or

`end`

EXCEPTION: Short lists whose conclusion is evident from the display format need not be annotated in this way.

REFERENCE: BB 1.9.7.

SEE ALSO: 4.2•7.

2.7.2 Display Control - Framing

•6 ▶ **Numbering Display Pages**

When display output is more than one page, annotate each page to indicate display continuation.

EXAMPLE: The phrase `page x of y` is commonly used for this purpose.

COMMENT: When a display extends over just a few pages, and when a user is not expected to care about any **particular** page, then it may be sufficient to identify the pages

```
first
continued
last
```

rather than assigning them numbers.

COMMENT: A recommended format is to identify pages by a note immediately to the right of the display title. With such a consistent location, the page note might be displayed in dimmer characters. Leading zeros should **not** be used in the display of page numbers.

COMMENT: If a large display output is viewed by continuous panning/scrolling rather than by discrete paging, then some other means must be found to label that portion of the display which is currently visible. Some sort of graphic indicator might be inset at the margin of the display frame to suggest current location.

REFERENCE: MS 5.15.3.1.12; PR 4.5.5, 4.10.4.

SEE ALSO: 2.5•4, 4.2•7.

•7 **Consistent Orientation - Panning vs. Scrolling**

Adopt a consistent orientation for display framing throughout interface design, so that users can either 1) conceive the display frame as a window moving over a fixed array of data, here called "panning", or 2) conceive data as moving behind a fixed display frame, commonly called "scrolling".

COMMENT: Ideally a consistent orientation for display framing would be maintained across all systems. Certainly that orientation should be consistent within any one system.

COMMENT: A user can adapt to either concept, if it is maintained consistently. Both concepts have some precedent in experience. Moving a camera across a fixed scene illustrates panning. Moving a specimen beneath the fixed eyepiece of a microscope illustrates scrolling. Tests seem to indicate that panning is the more natural concept for inexperienced users, causing fewer errors, and hence is the preferred option when other considerations are equal.

REFERENCE: Bury, Boyle, Evey and Neal, 1982.

▸ Panning with Free Cursor Movement •8

In applications where a user can move a cursor freely about a page of displayed data, adopt panning rather than scrolling as the conceptual basis of display framing.

EXAMPLE: Full-screen editing is a common application involving free cursor movement.

COMMENT: Since displayed data will be perceived as fixed during free cursor movement, considerations of **joint** compatibility suggest that displayed data remain conceptually fixed during movement of a display frame or window. Indeed, it might be possible to use the same arrow-labeled function keys to control both cursor movement and panning.

REFERENCE: Morrill and Davies, 1961.

SEE ALSO: 1.3•3.

Functional Labeling for Display Framing •9

When a user will access different systems with different conventions for panning or scrolling, in user instructions, key labels, etc., always refer to display framing in functional terms (e.g., "forward" and "back", or "next" and "previous") and avoid wording that implies spatial orientation (e.g., "up" and "down").

COMMENT: Control of display framing functions might be implemented by keys marked with arrows, to avoid verbal labels altogether.

COMMENT: Note that "forward" and "back" are potentially ambiguous because of the contradictory use of those words in referring to movement within books.

▸ Labeling Panning Functions •10

When a panning orientation is maintained consistently, choose names for display framing functions that refer to movement of the display frame (or window) and not to movement of the displayed data.

EXAMPLE: In this case, the command "Up 10" should mean that the display frame will move up ten lines, with the effect that ten lines of previous data will appear at the top of the display, and ten lines of subsequent data will disappear at the bottom.

2.7.2 Display Control - Framing

•11 **▶ Labeling Scrolling Functions**

When a scrolling orientation is maintained consistently, choose names for display framing functions that refer to movement of the data being displayed, and not to movement of the display frame (or window).

EXAMPLE: In this case, the command "Up 10" should mean that displayed data will move up ten lines behind the (conceptually fixed) display frame, with the effect that ten lines of previous data will disappear from the top of the display, and ten lines of subsequent data will appear at the bottom.

REFERENCE: EG 2.3.16.

•12 **Panning for Flexible Display Framing**

When a display exceeds the capacity of a single frame, in terms of the required extent and detail of coverage, consider providing users a capability to pan the display frame over the data in order to examine different areas of current interest.

COMMENT: A panning capability, sometimes called display "offset", can allow a user to move continuously over an extended display in any desired direction, without encountering any internal boundaries imposed by predefined display framing. Panning control might be accomplished by some continuous-action device such as a joystick, perhaps "dragging" a displayed framing element and then requesting display regeneration. There is some risk that a user might become disoriented in this process.

COMMENT: An alternative approach is to define discrete pages or sections of a large display, assign those sections identifying labels, and allow a user to specify directly which section should be displayed. This approach requires the user to pay specific attention to the labeling of different sections, which extra effort may help maintain orientation to the display as a whole. One risk in this approach, for a continuous display such as a map, is that an area of interest might be at a boundary where none of the sections provide adequate coverage. Thus some degree of overlapped coverage might be needed at the boundaries of displayed sections.

COMMENT: For some applications, it might prove helpful to allow a user to specify a particular location (such as a point on a map) and then automatically offset the display frame to be centered around that location.

SEE ALSO: 2.4.6•3, 2.4.8•10.

Zooming for Display Expansion •13

When a user may need to perceive displayed data relations more accurately, or to view data in pictures, diagrams, maps, etc. in greater detail, provide a zooming capability that allows the user to expand the display of any selected area.

COMMENT: Zooming can increase display spacing among crowded data items so that they can be perceived better. Thus an air traffic controller might expand a portion of a situation display to see more clearly the spacing of converging tracks that threaten a collision.

COMMENT: Zooming can increase the degree of detail, i.e., can add data to a display. Thus a user might expand a city map to see detailed road structures that are not shown in a small-scale map. When used this way, a zooming capability implies that data can be "layered" hierarchically at different levels of aggregation, which may require complex data files and data management techniques.

COMMENT: Zooming might be implemented as a continuous function, by which a display can be expanded to any degree, analogous to a continuous panning capability. Or zooming might be implemented in discrete increments, as in increasing the magnification of an optical instrument to x2, x4, etc. Incremental zooming, with abrupt changes in display scale, may tend to disorient a user, but might prove acceptable in some applications.

REFERENCE: Foley and Van Dam, 1982.

SEE ALSO: 2.4•15.

▶ Show Changing Scale •14

When a display has been expanded from its normal coverage, provide some scale indicator of the expansion factor.

EXAMPLE: A linear indicator of current map scale might be shown in the margin, or perhaps simply a numeric indication of the display expansion factor (e.g., x4).

COMMENT: In many applications it may be helpful to show the scale even for a display with normal, unexpanded coverage.

SEE ALSO: 2.4•16.

2.7.2 Display Control - Framing

•15 **Show Overview Position of Visible Section**

When a display has been panned and/or expanded from its normal coverage, provide some graphic indicator of the position in the overall display of the currently visible section.

EXAMPLE: In a corner of the display frame the computer might show a rectangle representing the overall display, in which a smaller rectangle is placed to indicate the position and extent of the currently visible portion of that display.

COMMENT: A graphic indication of the current coverage of a panned or expanded display will provide some visual context to help a user maintain a conceptual orientation between the visible part and the whole display from which that part has been excerpted.

COMMENT: In some special applications it may be possible to provide a user with two separate display screens, one to show an overview for general reference, and the other to show an expanded portion of that overview display.

REFERENCE: Foley and Van Dam, 1982.

SEE ALSO: 2.4•17, 2.4.8•11.

•16 **Framing Integrally for All Data**

Ensure that framing functions perform integrally so that panning and/or zooming will affect all displayed data in the same way.

EXAMPLE: On a situation display, zooming should expand background data such as geographic boundaries to the same scale as the expansion of overlaid "active" data.

•17 **Return to Normal Display Coverage**

If a user is allowed to pan over an extended display, or zoom for display expansion, provide some easy means for the user to return to normal display coverage.

EXAMPLE: Return to normal display coverage might be accomplished by a function key labeled RETURN, or perhaps RESET.

COMMENT: A user who has panned to some special area in an extended display, or who has expanded a display to examine some particular section in detail, may suddenly need to return quickly to normal display coverage. Perhaps the user has received an alerting message requiring attention to another portion of the display. Quick return to normal coverage will allow the user to re-establish a familiar orientation to the display as a whole.

COMMENT: Here normal coverage might always be defined as that data coverage shown when a display was initially generated. Or it might be desirable in some instances to allow a user to specify what is to be considered normal coverage for any particular display.

Update refers to the regeneration of displayed data, by user request or automatically, to show current changes.

Automatic Display Update •1

When displayed data are changing as a result of external events, a user should be able to request automatic update (computer regeneration) of changed data, and be able to control the update rate.

EXCEPTION: In an operations monitoring task, requirements may dictate the circumstances and rate of display updating.

REFERENCE: MS 5.15.3.4.2.

Highlighting Changed Data •2

When data have changed following automatic display update, consider highlighting those data changes temporarily.

EXAMPLE: A change in a critical data item might be highlighted with reverse video, or might be marked with a blinking symbol.

COMMENT: The desirable interval for highlighting changed data will depend on the importance of those data. If data changes imply a need for special user attention, then highlighting might continue until the user takes some specific acknowledgement action. Otherwise, highlighting might be removed after several seconds, or might continue until a user takes some other control action.

SEE ALSO: 2.6•1

Readability of Changing Data •3

If users must accurately read changing data values, ensure that those data are displayed long enough to be read.

COMMENT: A current design standard specifies that for accurate reading, data should be displayed in a fixed position and updated no more than once per second. In some applications, however, a slower update rate may be required. When in doubt, test user performance with prototype displays to determine appropriate update rates.

COMMENT: If users need only to monitor general trends in changing data values, and do not need to take exact readings, somewhat faster update rates may be acceptable. Again, prototype testing will sometimes be needed to determine appropriate update rates.

REFERENCE: MS 5.15.3.4.1.

2.7.3 Display Control - Update

•4 **Visual Integration of Changing Graphics**

If a user must visually integrate changing patterns on a graphic display, update the data at a rate appropriate to human perceptual abilities for that kind of data change.

EXAMPLE: Effective display of changing radar data for track detection and monitoring requires repeated, sequential output of stored data frames, and the timing of successive frames is critical for optimizing pattern perception.

COMMENT: Slowly developing patterns may be seen more easily with time compression, i.e., with rapid display of sequentially stored data frames. Fast changing data may require time expansion, i.e., slowed output, to aid pattern perception. For critical applications, experiment with prototype displays to determine proper timing.

COMMENT: In some applications it may help to allow a user to control the speed for update of displayed data.

COMMENT: In applications where the timing of display update is variable, it may help to indicate the currently selected time scale on the display.

COMMENT: Similar considerations apply to auditory displays, where speeding or slowing sound signals may aid pattern recognition.

REFERENCE: MS 5.15.3.6.3; Chao, 1985; Foley and Van Dam, 1982.

•5 **Display Freeze**

When displayed data are automatically updated, allow users to stop the process (by "freeze" or "stop action") at any point, in order to examine changed data more deliberately.

EXCEPTION: For an operations monitoring task, requirements may dictate that current data changes be continuously viewed; in such a case, display freeze might be useful during some subsequent playback of recorded data for purposes of operational analysis.

COMMENT: For some applications, it might also prove helpful if a user could step incrementally forward or back in a time sequence, in order to examine data changes frame by frame.

REFERENCE: MS 5.15.3.4.3.

•6 **▶ Labeling Display Freeze**

When a display has been frozen, annotate that display with some appropriate label to remind users of its frozen status.

REFERENCE: MS 5.15.3.4.4.

SEE ALSO: 4.4•13.

▶ Signaling Changes to Frozen Data •7

When a display being updated in real time has been frozen, warn users if some significant (but not displayed) change is detected in the computer processing of new data.

SEE ALSO: 4.4•13.

▶ Resuming Update After Display Freeze •8

When a display being updated in real time has been frozen, and then a user elects to resume update, resume display update at the current real-time point unless otherwise specified by the user.

COMMENT: In some applications, a user might wish to resume display update at the point of stoppage, and so display change would thenceforth lag real-time data change. Or perhaps a user might choose to see a speeded "replay" of interim changes to regain current display status. In practice, however, such options might risk confusion.

REFERENCE: MS 5.15.3.4.3.

Prediction Display •9

To help a user understand and respond effectively to complex data changes, consider displaying a prediction of future data states based on computer analysis of an appropriate model of the data dynamics.

EXAMPLE: Prediction display might be used to aid the control of any rapidly changing process involving complex dynamics, such as depth control for a high-performance submarine.

COMMENT: The concept of prediction display extends the practice of dynamic display update, from simply showing recent and current data states, to anticipate future changes in data. In effect, a computer can iterate in "fast time" changes in a dynamic data model, and then display its current prediction of future real-time changes. The usefulness of such prediction will depend upon the accuracy of the underlying data model. As it happens, where prediction display can help users, as in complex vehicular control or process control applications, the dynamics of process change are often defined sufficiently well to permit valid computer modeling.

COMMENT: A consistent logic should be adopted for computer modeling and prediction in relation to possible user action. For dynamic control appplications, one feasible logic is for a computer to predict and display the future consequences if the user persists in the current control action. As an alternative, a computer might predict and display the consequences if the user were to cease any control action. Either logic could prove helpful. But whichever is adopted should be used consistently.

COMMENT: Prediction displays should be formatted to distinguish clearly between actual current data and extrapolated future data.

2.7.4 Display Control - Suppression

Suppression refers to user control of display coverage by temporary deletion of specified data categories.

•1 **Temporary Suppression of Displayed Data**

When standard data displays are used for special purposes, allow users to temporarily suppress the display of data not needed for the current task.

COMMENT: Data selections made originally for one purpose may not be appropriate for another. When task requirements shift rapidly, it may be more efficient to suppress temporarily the display of unneeded data categories, rather than to regenerate a display with different selection criteria.

SEE ALSO: 2.0•1.

•2 ▶ **Labeling Display Suppression**

When data have been suppressed from a display, annotate the display with some appropriate label to remind users that data have been suppressed.

•3 ▶ **Signaling Changes to Suppressed Data**

When data have been suppressed from a display, warn users if some significant (but not displayed) change is detected in the computer processing of new data.

•4 ▶ **Resuming Display of Suppressed Data**

When data have been suppressed from a display, provide users with some means to quickly restore the display to its complete, originally generated form.

COMMENT: If a function key is used to restore suppressed data, that key action should have no other consequences. For instance, if a user must press RETURN to restore suppressed data, that key should only restore data and should **not** also move a displayed cursor to some other position.

COMMENT: In some applications, it may be desirable to restore suppressed data automatically, after expiration of a predetermined time-out, rather than relying on a user to remember to do it.

Window overlays can be temporarily added to a display to show requested data, menus, user guidance, etc.

Temporary Window Overlays •1

When it is necessary to add requested data or other features temporarily to a current display, consider providing window overlays for that purpose.

EXAMPLE: On a map display, if a user requests detailed data about a particular location, the computer might display an overlaid window with tabular or textual description, which could later be removed by user action.

COMMENT: Here temporary window overlays should be distinguished from the more stable "windows" consistently provided in display formatting, such as the various defined display areas that might be dedicated to showing the display title, a date-time group, user prompting, a composed control entry, an error message, etc.

COMMENT: Window overlays can be used to show data of temporary or extraneous interest. By contrast, if a set of data must be viewed continuously or in an integrated display with other data, then that data set should be available as a selectable data category.

SEE ALSO: 2.7.1•5, and Section 2.5.

Predefined Windows •2

When the value of particular window overlays can be determined during interface design, those overlays should be predefined and offered to users under computer control.

EXAMPLE: A menu of currently appropriate control options might be superimposed on a current display by user selection of the displayed menu title.

COMMENT: The aim here is to allow a user to select window overlays that have already been designed, rather than having to specify a desired window from scratch. User display control should be kept as simple as possible, so that a user can spend time assimilating data instead of manipulating the display of data.

2.7.5 Display Control - Window Overlays

•3 **User-Specified Windows**

When the need to view several different kinds of data jointly cannot be determined in advance, allow a user to specify and select separate data windows that will share a single display frame.

COMMENT: Users may abuse such a capability for arbitrary window definition, adding so many windows to a display that the resulting hodgepodge defies interpretation. A designer can do little to prevent that. However, the designer can try to ensure that the means for window creation and control are made as efficient as possible.

COMMENT: Depending upon user needs (and system capability), data windows might appear simultaneously as segments of a joint display, might be overlaid in varying degrees so as to obscure one another, or might be displayed sequentially at the user's option. In the latter condition, multiple display windows will differ little from multiple display pages, except perhaps in speed of sequential access.

COMMENT: This recommendation assumes that it is mostly **data** overlays which a user will want to create. Other kinds of window overlays would usually be offered under computer control, such as those providing error messages and other forms of user guidance. It is possible, however, that a user might wish to define certain kinds of non-data windows, such as an overlay of "favorite" menu options, or an overlaid guidance "memo" composed for the user's own purposes. Perhaps a user should be allowed to create those kinds of window overlays as well.

SEE ALSO: 2.5•8.

•4 **Consistent Window Control**

Ensure that the means provided users to control window overlays operate consistently from one display to another for each type of overlay.

COMMENT: Control of predefined windows may simply involve "opening" and "closing" them, by selection of displayed option labels or function keys. Control of user-defined windows may require user specification of window contents, window size and positioning on the display. Such window control must be learned by a user, and consistent design of control logic will aid that learning.

COMMENT: Some advocates of window overlays predict that standard methods of window control will become part of the basic support software for user interface design.

REFERENCE: Foley and Van Dam, 1982.

Easy Suppression of Window Overlays •5

Provide an easy means for a user to suppress the display of window overlays.

EXAMPLE: A requested guidance overlay might be removed by user selection of a RETURN function key; a menu overlay displayed under computer control might disappear automatically following user selection of an option from that menu.

SEE ALSO: Section 2.7.4.

Labeling Windows •6

When a user can select predefined window overlays, assign to each overlay an identifying label.

COMMENT: Labeling window overlays may help users request and recognize them, in the same way that display labeling can aid display selection.

SEE ALSO: 2.7.1•2.

Indicate Active Window •7

If several window overlays are displayed at once, indicate to the user in which window (if any) an action can currently be taken.

EXAMPLE: A prominent cursor might be displayed in the currently active window, or perhaps the displayed border of an active window might be highlighted in some way.

COMMENT: Addding window overlays to a display can increase the conceptual complexity of control actions as well as the difficulty of data assimilation. Consider a case in which several windows are shown, including some menu overlays and some data windows. There it is important to indicate to a user which window is currently "active", i.e., whether the next action will result in a menu selection or a data change.

SEE ALSO: 2.6•1.

▶ **Easy Shifting Among Windows** •8

If several window overlays are displayed at once, provide some easy means for a user to shift among them to select which window shall be currently active.

COMMENT: The most direct method might be to allow a user to select a window by pointing anywhere within its displayed borders, but that action might be confused with the selection of a particular item within the window. Some designers provide a special symbol for window selection in the border itself.

2.7.5 Display Control - Window Overlays

•9 **Consistent Control Within Windows**

When control actions such as command entry may be taken by a user working within a window overlay, ensure that those control actions will be consistent from one window to another.

EXAMPLE: Cursor positioning controls and data editing capabilities should operate consistently within all windows.

COMMENT: If controls in one window operate differently than in another, user confusion will be unavoidable.

•10 **Nondestructive Overlay**

When a window overlay temporarily obscures other displayed data, ensure that the obscured data are **not** permanently erased but will reappear if the overlay is later removed.

SEE ALSO: 2.7.1•10.

Design change of software supporting data display functions may be needed to meet changing operational requirements.

Flexible Design for Data Display •1

When data display requirements may change, which is often the case, provide some means for users (or a system administrator) to make necessary changes to display functions.

COMMENT: Display characteristics that may need to be changed include those represented in these guidelines, namely, the types of data that can be displayed, formatting and coding logic, and the capabilities offered for display control.

COMMENT: Many of the preceding guidelines in this section imply a need for design flexibility. Much of that needed flexibility can be provided in initial interface design. Some guidelines, however, suggest a possible need for subsequent design change, and those guidlines are cited below.

SEE ALSO: 2.0•2, 2.0•8, 2.0•11, 2.5•8, 2.7.1•1, 2.7.1•3.

Section 3
SEQUENCE CONTROL

SECTION 3

SEQUENCE CONTROL

Sequence control refers to user actions and computer logic that initiate, interrupt, or terminate transactions. Sequence control governs the transition from one transaction to the next. General design objectives include consistency of control actions, minimized need for control actions, minimized memory load on the user, with flexibility of sequence control to adapt to different user needs. Methods of sequence control require explicit attention in interface design, and many published guidelines deal with this topic.

The importance of good design for controlling user interaction with a computer system has been emphasized by Brown, Brown, Burkleo, Mangelsdorf, Olsen and Perkins:

> One of the critical determinants of user satisfaction and acceptance of a computer system is the extent to which the user feels in control of an interactive session. If users cannot control the direction and pace of the interaction sequence, they are likely to feel frustrated, intimidated, or threatened by the computer system. Their productivity may suffer, or they may avoid using the system at all.
>
> (1983, page 4-1)

Complete user control of the interaction sequence and its pacing is not always possible, of course, particularly in applications where computer aids are used for monitoring and process control. The actions of an air traffic controller, for example, are necessarily paced in some degree by the job to be done. As a general principle, however, it is the user who should decide what needs doing and when to do it.

A fundamental decision in user interface design is selection of the dialogue type(s) that will be used to implement sequence control. Here "dialogue" refers to the sequence of transactions that mediate user-system interaction. Interface design will often involve a mixture of two or more dialogue types, since different dialogues are appropriate to different jobs and different kinds of users. Recognition of appropriate dialogue types at the outset of system development will facilitate the design of user interface software and help ensure the effectiveness of system operation.

The selection of dialogue types based on anticipated task requirements and user skills seems straightforward, at least for simple cases. Computer-initiated question-and-answer dialogues are suited to routine data entry tasks, where data items are known and their ordering can be constrained; this type of dialogue provides explicit prompting for unskilled, occasional users. Form-filling dialogues permit somewhat greater flexibility in data entry, but may require user training. When data entries must be made in arbitrary order, perhaps mixed with queries as in making airline reservations, then some mixture of function keys and coded command language will be required for effective operation, implying a moderate to high level of user training.

Introduction

One important aspect of dialogue choice is that different types of dialogue imply differences in system response time for effective operation. In a repetitive form-filling dialogue, for example, users may accept relatively slow computer processing of a completed form. If the computer should take several seconds to respond, a user probably can take that time to set one data sheet aside and prepare another. But several seconds delay in a menu selection dialogue may prove intolerable, especially when a user must make an extended sequence of selections in order to complete an action.

To categorize these differences, the estimated requirement for user training and for system response time is given below for eight dialogue types.

Dialogue Type	Required User Training	Tolerable Speed of System Response
Question and Answer	Little/None	Moderate
Form Filling	Moderate/Little	Slow
Menu Selection	Little/None	Very Fast
Function Keys	Moderate/Little	Very Fast
Command Language	High	Moderate/Slow
Query Language	High/Moderate	Moderate
Natural Language	Moderate/Little	Fast
Graphic Interaction	High	Very Fast

The general requirements estimated in this table may vary, of course, with any specific system design application. As an example, graphic interaction is judged here to require a high degree of user training. That would surely be true of systems providing a full range of graphic functions. But in other applications some simple forms of graphic interaction, such as iconic representation of menu options, might actually serve to reduce the need for user training.

This categorization of dialogue types has been adopted from that proposed by Ramsey and Atwood (1979). Each of these dialogue types is considered in the guidelines presented here. But much pertinent material will be found elsewhere in these guidelines. Thus form filling is considered here as a dialogue type for sequence control, but is treated elsewhere as a means of data entry (Section 1.4) and of data display (Section 2.2). Graphic interaction, considered here as a dialogue type for sequence control, is also treated extensively as a means of data entry (Section 1.6) and data display (Section 2.4).

One might speculate whether some other type of sequence control dialogue, beyond those listed here, could become commonplace in the future. Imagine an application where a user interacts with a so-called "expert" computer system, with the locus of control in query and reply shifting back and forth. Would that represent merely a combination of the various dialogue types considered here? Or should we define some new type of interaction called "mutual consultation" to deal more effectively with that situation? This remains an open question.

Regardless of the dialogue type(s) chosen, providing context, consistency and flexibility will be important for sequence control as it is for other aspects of user interface design. Several guidelines proposed here deal explicitly with the need to define and maintain context for users.

With regard to consistency of sequence control, it should be emphasized that users of information systems regard their computer as a tool necessary to perform a job. As a tool, they expect the computer to perform efficiently, reliably, and predictably. They will not regard the computer as an intriguingly unpredictable toy with which to play games. Elements of surprise that might be entertaining in a game will be frustrating in a tool.

Neither will users want to regard their computer as a puzzle to be solved, a challenging device whose intricacies promise pleasure in mastery. Where a programmer might be intrigued by the problems of instructing a computer to perform a difficult task, an ordinary user of the system may merely be irritated by the complexity of a computer tool. Where smart shortcuts are provided to perform particular tasks in particular ways, the ordinary system user may resent the extra learning involved, and the extra memory load, rather than appreciate the elegance of saving keystrokes.

This argument for consistent control rather than smart shortcuts has been made elsewhere (e.g., Reisner, 1981) but merits continual repetition. Perhaps the most frequent mistake made by designers of user interface software is to provide smart shortcuts instead of consistent control procedures. In every instance, the designer's intent is to help users — by shortening a particular command, by saving a logically redundant keystroke, or by making sequence control more efficient for a knowledgeable user with perfect memory. But no real users fit that description. Real users depend upon consistent interface design to set practical limits on what they must learn and remember about their computer tools.

In accord with this argument, many of the guidelines proposed here deal in some way with the need to provide consistent logic for sequence control. A consistent interface design — where actions are all taken in the same way, where displayed control options are all formatted in the same way, where error messages are all worded in the same way, and so on — may seem dull to its designers. It may even seem dull to some of its users. But it should prove easy to learn. Smart shortcuts, i.e., the special design features that can make particular control actions more efficient, should be provided only as optional extras, not needed by novice users but offering some flexibility for experienced users.

Ideal flexibility would permit experienced users to undertake whatever task or transaction is needed, at any time. Although this may not always prove feasible, the interface designer should try to provide the maximum possible user control of the on-line transaction sequence. As a simple example, a user who is scanning a multipage data display should be able to go either forward or back at will. If interface software only permits stepping forward, so that users must cycle through the entire display set to reach a previous page, that design is inefficient. Users should also be able to interrupt display scanning at any point to initiate some other transaction. Such simple flexibility is relatively easy for the designer to achieve, and indeed is commonly provided.

Introduction

More difficult are transactions that involve potential change to stored data. Here again users will need flexibility in sequence control, perhaps wishing to back up in a data entry sequence to change previous items, or to cancel and restart the sequence, or to end the sequence altogether and escape to some other task. The interface designer can provide such flexibility through use of suspense files and other special programmed features. That flexibility may require extra effort from the software programmer. But that extra effort is made only once, and is a worthwhile investment on behalf of future users who may interact with their computer system for months or even years.

Of course, flexibility of sequence control has pitfalls. Just as users can make mistakes in data entry, so also will users make mistakes in sequence control. The interface designer must try to anticipate user errors and ensure that potentially damaging actions are difficult to take. In most data entry tasks, for example, simple keying of data items should not in itself initiate computer processing. The user should have to take some further, explicit action to ENTER the data. The interface logic should be designed to protect the user from the consequences of inadvertently destructive actions. Any large-scale erasure or deletion of data, for example, should require some sort of explicit user confirmation, being accomplished as a two-step process rather than by a single keystroke. (This provides a software analogy to the physical barriers sometimes used to protect critical hardware controls from accidental activation.) Some well-designed systems go a step further and permit the user to reverse (UNDO) a mistaken action already taken.

One form of flexibility frequently recommended is the provision of alternate modes of sequence control for experienced and inexperienced users. In a command-language dialogue, optional guidance might be provided to prompt a beginner step by step in the composition of commands, whereas an experienced user might enter a complete command as a single complex input. Some such flexibility in the user interface is surely desirable — so that the computer can interpret halting, stepwise control inputs, as well as fluent, coherent commands.

More generally, however, it may be desirable to include redundant modes of sequence control in user interface design, perhaps involving combinations of different dialogue types. As an example, menu selection might be incorporated to provide easy sequence control for beginners, but every display frame might also be formatted to include a standard field where an experienced user could enter complete commands more efficiently. Examples of that approach have been provided by Palme (1979).

Another way to provide flexibility in sequence control is through specific tailoring of display formats. Consider, for example, a menu selection dialogue in which sequence control is exercised through lightpen selection among displayed control options. For any particular display frame it might be possible to display just three or four options most likely to be selected by a user at that point in the task sequence, plus a general purpose OPTIONS selection that could be used to call out a display of other (less likely) commands. Thus, on the first page of a two-page display set, one of the likely commands would be NEXT PAGE; but on the second page that command would be replaced by its more likely complement, PREV PAGE.

This approach illustrates two design ideas. The first comes close to being a general principle for sequence control: make the user's most frequent transactions the easiest to accomplish. The second idea is the reliance on context to improve flexibility. These general ideas concerning sequence control are reflected in the specific design guidelines proposed in the following pages.

Objectives:

Consistency of control actions

Minimal control actions by user

Minimal memory load on user

Compatibility with task requirements

Flexibility of sequence control

Functions

Sequence control refers to user actions and computer logic that initiate, interrupt, or terminate transactions.

Flexible Sequence Control •1

Provide flexible means of sequence control so that users can accomplish necessary transactions involving data entry, display, and transmission, or can obtain guidance as needed in connection with any transaction.

EXAMPLE: In scanning a multipage display the user should be able to go forward or back at will. If user interface design permits only forward steps, so that the user must cycle through an entire display series to reach a previous page, that design is deficient.

COMMENT: Necessary transactions should be defined in task analysis prior to software design.

REFERENCE: PR 4.0.

Minimal User Actions •2

Ensure that control actions are simple, particularly for real-time tasks requiring fast user response; control logic should permit completion of a transaction sequence with the minimum number of actions consistent with user abilities.

EXAMPLE: A user should be able to print a display by simple request, without having to take a series of other actions first, such as calling for the display to be filed, specifying a file name, then calling for a print of that named file.

EXAMPLE: For long, multipage displays, it should be possible to request a particular page directly, without having to take repetitive NEXT PAGE or PREV PAGE actions.

EXCEPTION: A destructive action will be less likely to be taken by mistake, if it is designed to be different or distinctive, requiring **extra** user actions.

COMMENT: Shortcuts via direct commands should allow experienced users to by-pass intervening steps that may help beginners. The computer should be programmed to handle automatically any intervening processing that may be required, informing the user what has been done if that becomes necessary (as in the case of a detected error).

REFERENCE: BB 2.4.1, 4.5; MS 5.15.4.6.4.

SEE ALSO: 4.0•24.

•3 **Control Matched to User Skill**

Ensure that the means of sequence control are compatible with user skills, permitting simple step-by-step actions by beginners, but permitting more complex command entry by experienced users.

COMMENT: Most systems will have users with varying levels of experience. Any particular user may become more expert with increasing experience, or perhaps less expert after a long period of disuse. Accommodating users of varying expertise will usually require a mixture of different dialogue types, with some means for smooth transition from one mode of dialogue to another. For instance, as a user comes to learn menu codes, s/he might be allowed to enter those codes without necessarily displaying a menu, i.e., those codes might also serve as commands.

REFERENCE: BB 4.5; Badre, 1984; Gilfoil, 1982.

SEE ALSO: 4.4•31, and Section 3.1.

•4 **User Initiative in Sequence Control**

Allow users to take initiative and control their interaction with the computer; try to anticipate user requirements and provide appropriate user control options and computer responses in all cases.

COMMENT: In most applications, a user should be able to interrupt or terminate any transaction once it has been initiated (see Section 3.3). Users will sometimes change their minds and decide that an initiated transaction is not what was wanted after all.

COMMENT: Software logic should be "bulletproofed" to anticipate **every** possible action by a user, no matter how improbable, providing an appropriate computer response for random (or even malicious) inputs as well as for correct entries and likely errors. In particular, a dialogue should never reach a dead end with no further action available to the user. If a user makes an entry inappropriate to current processing logic, the computer should simply display an advisory message that the input cannot be recognized and indicate the available options as to what can be done next.

REFERENCE: BB 4.2.5.1; PR 2.2.

SEE ALSO: 4.4•5.

Control by Explicit User Action •5

Allow users to control transaction sequencing by explicit action; defer computer processing until an explicit user action has been taken.

EXAMPLE: When a user is keying an extended data entry, the computer should **not** interrupt the user to require immediate correction of any entry error, but instead should wait for the user's ENTER action.

EXAMPLE: When a user is composing a command to accomplish some transaction, the computer should **not** interrupt the user by responding as soon as it recognizes a partial entry, but instead should wait for the user's ENTER action.

EXCEPTION: In automated process control applications, emergency conditions may take precedence over current user transactions, and a computer-generated warning might interrupt user actions.

EXCEPTION: In routine, repetitive data entry transactions, successful completion of one entry may lead automatically to initiation of the next, as in keying ZIP codes at an automated post office.

COMMENT: If the computer interrupts a user, it pre-empts the initiative in sequence control, in effect forcing the user into an error correction (or some other) sequence conceived by the interface designer, and not necessarily a sequence that would be chosen by the user.

COMMENT: Some interface designers devise computer interruptions that they suppose will help a user, as when they program a computer to complete a partial command automatically as soon as it recognizes the user's intended entry. Many users, however, will find unexpected computer interruptions more disconcerting than helpful, and may become confused at least momentarily as to just what they **had** intended.

COMMENT: In general, computer detection of problems with current user entries can be negotiated at the conclusion of a transaction, before it is implemented. Nondisruptive alarms or advisory messages can be displayed to report computer monitoring of external events so that the user can choose when to deal with them.

SEE ALSO: 1.0•9, 1.1•4, 1.4•1, 4.0•2, 6.0•9, 6.3•5.

•6 **Consistent User Actions**

Ensure that sequence control actions are consistent in form and consequences; employ similar means to accomplish ends that are similar, from one transaction to the next, from one task to another, throughout the user interface.

COMMENT: In particular, there should be some standard, consistent routine for a user to initiate and terminate the transaction sequences that comprise different tasks. Do **not** require users to learn different command names to terminate different tasks, or remember to terminate one task by command and another by function key.

COMMENT: Interface designers may sometimes be tempted to deviate from consistent control syntax in order to minimize keystrokes in a particular transaction. Or designers may wish to add a new, improved syntax for functions added later in system development. Though such inconsistencies may in each case be intended to help users, they will make **all** functions more difficult for users to learn.

REFERENCE: Mooers, 1983; Reisner, 1981.

SEE ALSO: 4.0•1.

•7 **Logical Transaction Sequences**

When designing a sequence of related transactions for some information handling task, employ task analysis to ensure that those transactions will constitute a logical unit or subtask from a user's viewpoint, and to determine what control options users will need at any point.

COMMENT: A logical unit to the user is not necessarily the same as a logical unit of the computer software that mediates the transaction sequence. It might be, for example, that a user should enter ten items of data in a single transaction, because those data all come from one particular paper form, even though the computer will use five of those items for one purpose and five items for another in its subsequent internal processing.

REFERENCE: PR 5.1; Stewart, 1980.

SEE ALSO: 4.0•1.

•8 **Distinctive Display of Control Information**

Design all displays so that features relevant to sequence control are distinctive in position and/or format.

COMMENT: Relevant features include displayed options, command entry areas, prompts, advisory messages, and other displayed items (titles, time signals, etc.) whose changes signal the results of control entries.

SEE ALSO: 2.5•2, 4.0•6.

Displayed Context •9

If the consequences of a control entry will differ depending upon context established by a prior action, then display some continuous indication of current context for reference by the user.

EXAMPLE: If activating a DELETE key establishes a **mode**, so that subsequent selection of a PAGE key will erase a page of data rather than simply advancing to display the next page, then some indication of that established DELETE mode should be displayed to the user.

COMMENT: Do not rely on the user always to remember prior actions, nor to understand their current implications.

SEE ALSO: 4.4•13, and Section 3.4.

Consistent Terminology for Sequence Control •10

For instructional material, such as display labeling, on-line guidance and other messages to users, adopt consistent terminology to refer to sequence control.

EXAMPLE: Various words and phrases might be used, such as "control input", "command entry", "instruction", "request", "function call", etc. The practice adopted in these guidelines is to call general sequence control actions "control entry". More specific terminology is sometimes used here, such as "command entry" for keyed control entries composed by the user, "code entry" for keyed selections from displayed menus, etc.

SEE ALSO: 4.0•15.

▶ Congruent Names for Control Functions •11

When selecting names for sequence control functions, choose names that are semantically congruent with natural usage, especially for paired opposites.

EXAMPLE: If one function name is UP, then DOWN (rather than LOWER, say) should accomplish an opposite function; PULL should be reversed by PUSH; FORWARD by BACKWARD; RIGHT by LEFT; IN by OUT; etc.

COMMENT: A user who learns one function name will assume that s/he can accomplish an opposite function by using a semantically opposite name. One implication is that names for sequence control functions should not be chosen independently. Another implication is that understanding a user's natural vocabulary is important for selecting good function names.

REFERENCE: Carroll, 1982.

•12 ▶ **Upper and Lower Case Equivalent**

For interpreting user-composed control entries, treat upper and lower case letters as equivalent.

COMMENT: Users find it difficult to remember whether upper or lower case letters are required, and so the interface design should not try to make such a distinction.

SEE ALSO: 1.0•27, 1.3•10.

•13 ▶ **Wording Consistent with User Guidance**

Ensure that the wording and required format of control functions is reflected consistently in the wording of user guidance, including all labels, messages, and instructional material.

EXAMPLE:

(Good) To delete a paragraph, press DELETE and then PARAGRAPH.

(Bad) If a paragraph must be erased, press DELETE and then PARAGRAPH.

EXAMPLE: When the computer displays a file name, that name should be shown in a format that would be acceptable if the name were included in a command entry; do **not** display a capitalized name if the computer will not accept a capitalized entry.

EXAMPLE: If a user must complete a control form to specify printer settings, the words used as labels on that form should also be used in any error messages and HELP displays which may guide that process.

COMMENT: When selecting or composing control entries, a user will tend to mimic the vocabulary, format, and word order used in computer displays, including labels, error messages, HELP displays, etc. If displayed wording is consistent with required entries, a user will be more likely to make a correct entry on the first try.

COMMENT: Consistency in wording will be particularly helpful for dialogues based on constrained natural language. If a designer begins by determining which words and formats users are likely to choose naturally, and then reinforces that usage by incorporating such wording in user guidance, much of a user's interaction with the computer will be predictable. Therefore the "natural language" need not accommodate the full range of possible entries, but only those entries which users are likely to make.

REFERENCE: Good, Whiteside, Wixon and Jones, 1984; Mooers, 1983; Zoltan-Ford, 1984.

SEE ALSO: 2.0•7, 3.1.7•1, 4.0•18.

Feedback for Control Entries •14

Ensure that the computer acknowledges every control entry immediately; for every action by the user there should be some apparent reaction from the computer.

EXAMPLE: Execution of a requested transaction might produce an immediately apparent result, as when a user requests NEXT PAGE and the next page is displayed.

EXAMPLE: A message might indicate completion of a transaction, as when a user requests printout at a remote facility and the computer displays a confirming message

AIRFIELD file has been sent to printer.

EXAMPLE: A message might indicate that execution is in progress or deferred, as when a user enters data and the computer displays an interim message

AIRFIELD file is being updated.

EXAMPLE: A message might indicate that the control entry requires correction or confirmation, as when a user requests file display and the computer displays an error message

"AIRFELD" file not recognized.

COMMENT: In particular, the **absence** of computer response is **not** an acceptable means of indicating that a control entry is being processed.

COMMENT: "Immediately" as used in this guideline must be interpreted in relation to the response time requirements of different dialogue types.

REFERENCE: BB 4.3.1, 4.3.2; EG 4.2.5; MS 5.15.5.2, 5.15.5.3, 5.15.2.1.3; Foley and Van Dam, 1982.

SEE ALSO: 1.0•3, 1.0•12, 1.0•13, 4.2•1, 4.2•3, and Section 3.1.

▶ Indicating Completion of Processing •15

When processing in response to a control entry is lengthy, give the user some positive indication of subsequent completion, and appropriate related information.

COMMENT: If a user is currently involved in some new transaction, then completion of processing for a prior transaction should be indicated by nondisruptive display of an appropriate advisory message.

COMMENT: If the outcome of a completed transaction may imply the need for further user action, that should be indicated to the user.

REFERENCE: BB 4.3.1; MS 5.15.5.2.

SEE ALSO: 4.2•4.

•16

▶ Compatibility with User Expectations

Ensure that the results of any control entry are compatible with user expectations, so that a change in the state or value of a controlled element is displayed in an expected or natural form.

EXAMPLE: A control entry of NEXT PAGE should show the next frame of a current display, and should **not** jump off to some other internally defined "page" in the computer's data base.

EXAMPLE: When the completion of a control entry is indicated by a special function key, that key should be labeled ENTER (or some functionally equivalent word) and should result in computer acknowledgment of the entry.

COMMENT: Compatibility between user action and system response is an important concept in human engineering design. Interface designers should not assume that user expectations will match their own. User expectations can be discovered by interview, questionnaire, and/or prototype testing. Where no strong user expectations exist with respect to a particular design feature, then designers can help **establish** valid user expectations by careful consistency in interface design.

REFERENCE: MS 5.15.4.1.12; Smith, 1981b.

SEE ALSO: 1.0•10, 1.1•15, 1.1•19, 3.0•6, 4.2•1.

•17

User-Paced Sequence Control

Allow users to pace control entries, rather than requiring users to keep pace with computer processing or external events.

COMMENT: User pacing will let control entries be made in accord with a user's current needs, attention span, and time available.

COMMENT: When user-paced control does not seem feasible, as in critical process control applications, reconsider the general approach to task allocation and user interface design, perhaps providing greater system automation to ensure timely response.

SEE ALSO: 1.0•8.

Appropriate Computer Response Time •18

Ensure that the speed of computer response to user control entries is appropriate to the transaction involved; in general, the response should be faster for those transactions perceived by a user to be simple.

EXAMPLE: Computer response to a likely control entry, such as NEXT PAGE, should be within 0.5-1.0 second; response to other simple entries should be within 2.0 seconds; error messages should be displayed within 2-4 seconds.

COMMENT: Interface designers may need to consult with the intended system users to decide upon appropriate computer response times for different transactions.

REFERENCE: EG Tables 2-3; MS Table XXIX; Foley and Van Dam, 1982; Miller, 1968; Shneiderman, 1984.

SEE ALSO: 1.0•4, 4.2•2, 4.3•11.

Control Availability •19

Allow users to make control entries as needed; a sequence of control entries should **not** be delayed by delays in computer response.

COMMENT: It is recommended that control delays or lockouts not exceed 0.2 seconds. In some applications, however, longer delay may be tolerable, particularly if that has the effect of reducing variability in computer response time.

SEE ALSO: 1.0•4.

▶ Indicating Control Lockout •20

If control entries **must** be delayed pending computer processing of prior entries, then indicate that delay to the user.

EXAMPLE: If processing delay results in control lockout, that could be signaled by disappearance of the cursor from the display, or perhaps by a notable change in the shape of the cursor, accompanied by an auditory signal.

COMMENT: In some applications it may be desirable to ensure that the keyboard and other control devices are automatically locked until the user can begin a new transaction. This would be true when processing the current transaction will affect the results of subsequent user actions. In other applications, it may be possible to permit users to continue work while previous transactions are still being processed.

COMMENT: Deletion or change of a displayed cursor in itself may not be a sufficient indicator of keyboard lockout. Auditory signals will be particularly helpful to a skilled touch-typist, who may not look at the display when transcribing data entries.

COMMENT: Following control lockout, computer readiness to accept further entries should be indicated to the user.

SEE ALSO: 4.1•4.

•21 ▶ **Interrupt to End Control Lockout**

In situations where control lockout **does** occur, provide the user with an auxiliary means of control entry, such as a special function key, to abort a transaction causing extended lockout.

COMMENT: Such an interrupt capability will be especially helpful if a user recognizes that an error has been made and wants to stop an unneeded transaction, acting like an UNDO command.

COMMENT: Alternatively, for some transactions it may be helpful to design this interrupt as an END command that stops ongoing processing without canceling it. For example, if a user has asked the computer to scroll ahead in a long file display, that user may simply wish to stop at a certain point rather than returning to the beginning.

SEE ALSO: 3.3•6.

•22 **Control by Simultaneous Users**

When several users must interact with the system simultaneously, ensure that control entries by one user do not interfere with those of another.

COMMENT: This requires careful interface design for applications where joint, coordinated actions must be made by a group of users.

REFERENCE: MS 5.15.4.6.5.

SEE ALSO: 6.0•4.

Dialogue types for sequence control must be designed to match the needs of different tasks and different users

Dialogue Matched to Task and User •1

Consider task requirements and associated user characteristics when choosing dialogue type(s) and designing sequence control logic.

EXAMPLE: When untrained users must choose among a fixed set of options (as in the case of automated bank teller machines) labeled function keys might suffice for sequence control; when options may be chosen from a larger set (as in public information systems) menu selection will prove a more efficient dialogue type.

EXAMPLE: When users must make data and control entries in an arbitrary order, perhaps mixed with queries (as in making flight reservations when talking with a customer), then some mixture of function keys and command entries will be required for effective operation.

COMMENT: A simple dictum is, "Know the user." However, if user characteristics are variable, which is usually the case, then provide a variety of dialogue types based on analysis of **task** requirements.

COMMENT: Choice of dialogue type(s) is a fundamental decision in interface design. Designers should consider that decision carefully. A poor choice can detract seriously from system usability and will be difficult to change later.

REFERENCE: MS 5.15.4.1.8; Martin, 1973.

SEE ALSO: 3.0•3, 4.4•31.

Appropriate Computer Response Time •2

Ensure that the speed of computer response to user entries is appropriate to the type of dialogue; the response to menu selections, function keys, and most entries during graphic interaction should be immediate.

COMMENT: It is generally thought that maximum acceptable delay for computer response to menu selection by lightpen is 1.0 second; for key activation is 0.1 second; for cursor positioning by lightpen (as in graphic line drawing) 0.1 second.

COMMENT: If computer response time will be slow, consider choosing other dialogue types, such as command entry.

REFERENCE: EG Tables 2-3; MS Table XXIX; Miller, 1968.

SEE ALSO: 3.0•18, 4.2•2.

3.1.1 Dialogue Type - Question and Answer

Question-and-answer dialogues, where the computer poses questions for a user to answer, are suited to novice users.

•1 **Question-and-Answer Dialogue**

Consider question-and-answer dialogues for routine data entry tasks, where data items are known and their ordering can be constrained, where users will have little or no training, and where computer response is expected to be moderately fast.

EXAMPLE: In the automated collection of medical history data, a computer might follow contingent branching logic in posing questions for patients to answer.

COMMENT: Brief question-and-answer sequences can be used to supplement other dialogue types for special purposes, such as for LOG-ON sequences, or for resolving ambiguous control or data entries.

COMMENT: Where computer response to any single user entry may be slow, then the aggregate time required to process a series of questions and answers may be very slow. In such a case, consider form filling as an alternative dialogue type, where the user can enter a set of related "answers" as a single transaction.

•2 **Questions Displayed Singly**

In question-and-answer dialogues, display each question separately; do **not** require users to answer several questions at once.

COMMENT: A user may become confused in trying to deal with several questions at once, particularly if the number of questions is variable from one transaction to another.

•3 **Recapitulating Prior Answers**

When a series of computer-posed questions are interrelated, display answers to previous questions when those will provide context to help a user answer the current question.

COMMENT: Do **not** rely on a user to remember prior answers.

COMMENT: Another way to request a related series of user entries is to use a form-filling dialogue rather than question-and-answer.

SEE ALSO: 3.4•3.

•4 **Sequence Compatible with Source Documents**

When questions prompt entry of data from a source document, ensure that the question sequence will match the data sequence in the source document.

SEE ALSO: 1.4•25.

Form filling permits a user to enter a series of related data items or control options as a single transaction.

Form Filling for Data Entry •1

Consider form filling for tasks where some flexibility in data entry is needed, such as the inclusion of optional as well as required items, where users will have moderate training, and/or where computer response may be slow.

EXAMPLE: Form filling might be an appropriate dialogue type for a computer system that helped users calculate income tax obligations.

COMMENT: Specific recommendations for the design of form-filling dialogues are presented in Section 1.4 for data entry and in Section 2.2 for data display.

REFERENCE: MS 5.15.4.3.1.

SEE ALSO: Section 1.4, Section 2.2.

Form Filling for Control Entry •2

Consider form filling as an aid for composing complex control entries.

EXAMPLE: For a complex data retrieval request, a displayed form might indicate the various control parameters that could be specified.

EXAMPLE: For a print request, a displayed form might help a user invoke the various format controls that are available.

Defaults for Control Entry •3

Consider form filling as a means of displaying default values for the parameters in complex control entries.

COMMENT: Default parameters permit users to compose potentially complicated control entries by relatively simple actions. If defaults have been defined, they should be indicated to users. A displayed form permits a user to review (and confirm or change) default control values, just as a user might review displayed defaults for data entry.

COMMENT: When only a few control parameters are involved, it may be feasible simply to prompt users with guidance messages rather than by displaying a control form.

REFERENCE: EG 4.2.4.

SEE ALSO: 3.1.5•4.

•4 **Consistent Format for Control Forms**

Ensure that forms for control entry are consistent in format; their design should generally conform to guidelines for the design of data entry forms.

SEE ALSO: Section 1.4, Section 2.2.

Menu selection permits a user to specify control entries by pointing at displayed options or keying associated codes.

Menu Selection •1

Consider menu selection for tasks that involve choice among a constrained set of alternative actions, that require little entry of arbitrary data, where users may have little training, and where computer response is relatively fast.

EXAMPLE: Displayed menus are commonly used for function selection in text processing, in graphic interaction, and a multitude of other applications.

COMMENT: Lengthy menus are often formatted as separate displays. Task-specific menus, however, can sometimes be incorporated effectively along with data displays, to provide a short list of appropriate control options.

COMMENT: Menu selection is, of course, a generally good means for control entry by untrained users. Menus can be used in conjunction with other dialogue types, depending upon task requirements. Sometimes a menu selection might be clarified by a supplementary question-and-answer dialogue.

COMMENT: If display output is slow, as on a printing terminal or on an electronic display constrained by a low-bandwidth channel, it may be tiresome for a user to wait for display of menu options, especially when selections must be made from several sequentially displayed menus. Under those conditions, experienced users may wish to by-pass menu selections in favor of direct command entry.

REFERENCE: MS 5.15.4.2.1.

Single Selection Per Menu •2

Each menu display should permit only one selection by the user.

COMMENT: Novice users will be confused by any more complicated procedure, such as a "Chinese menu" requiring one choice from Column A, one from Column B, etc.

REFERENCE: PR 4.6.5.

•3 **Single-Column List Format**

When multiple menu options are displayed in a list, display each option on a new line, i.e., format the list as a single column.

EXCEPTION: Displaying options in several columns may be considered where shortage of display space dictates a compact format; if there are only a few options, those might be displayed in a single row.

EXCEPTION: An interesting exception could be made for hierarchic menus, where a high-level menu might be shown in the left column of a display, accompanied by a lower-level menu in the right column whose options change to relfect whatever selection is currently made from the high-level menu.

COMMENT: A single-column list format will aid scanning and assimilation of available options, especially for novice users.

REFERENCE: MS 5.15.4.2.7.

SEE ALSO: 2.1•20.

•4 **Menu Selection by Pointing**

When menu selection is the primary means of sequence control, and especially if choices must be made from extensive lists of displayed control options, permit option selection by direct pointing (e.g., by touch display, lightpen, etc.).

EXCEPTION: If a capability for direct pointing is **not** provided (e.g., if pointing involves separate manipulation of a mouse, or cursor positioning by key action), then for long menus it may prove faster to permit menu selection by keying associated option codes.

COMMENT: Pointing directly at a displayed option guarantees good display-control compatibility. Users do not have to note associated option codes and enter them by key actions.

REFERENCE: MS 5.15.2.5.1, 5.15.4.2.2; Shinar, Stern, Bubis and Ingram, 1985; Thompson, 1971.

SEE ALSO: 1.1•12.

•5 ▸ **Large Pointing Area for Option Selection**

If menu selection is accomplished by pointing, as on touch displays, design the acceptable area for pointing to be as large as consistently possible, including at least the area of the displayed option label plus a half-character distance around that label.

COMMENT: The larger the effective target area, the easier the pointing action will be, and the less risk of error in selecting a wrong option by mistake.

REFERENCE: BB 2.12; EG 2.3.13, 6.1.3.

SEE ALSO: 1.1•13.

▸ Dual Activation for Pointing •6

If menu selection is accomplished by pointing, provide for dual activation, in which the first action designates (positions a cursor at) the selected option, followed by a separate second action that makes an explicit control entry.

EXAMPLE: On a touch display, the computer might display a separate ENTER box that can be touched by a user to indicate that the cursor has been properly positioned.

COMMENT: The two actions of cursor placement and entering should be compatible in their design implementation. If the cursor is positioned by keying, then an ENTER key should be used to signal control entry. If the cursor is positioned by lightpen, provide a dual-action "trigger" on the lightpen for cursor positioning and control entry.

COMMENT: This recommendation for dual activation of pointing assumes that accuracy in selection of control entries is more important than speed. In some applications that may not be true. Interface design will involve a trade-off considering the criticality of wrong entries, ease of recovery from wrong entries, and user convenience in making selections.

REFERENCE: Foley and Wallace, 1974.

SEE ALSO: 1.0•9, 1.1•4, 3.0•5, 6.0•9.

Menu Selection by Keyed Entry •7

When menu selection is a secondary (occasional) means of control entry, and/or only short option lists are needed, then consider accomplishing selection by keyed entry.

COMMENT: An option might be selected by keying an associated code which is included in the displayed menu listing. Alternatively, if menu labels can be displayed near a screen margin, then an option might be selected by pressing an adjacent multifunction key.

•8 ▶ **Standard Area for Code Entry**

When menu selection is accomplished by code entry, provide a standard command entry area (window) where users enter the selected code; place that entry area in a fixed location on all displays.

COMMENT: In a customary terminal configuration, where the display is located above the keyboard, command entry should be at the bottom of the display, in order to minimize user head/eye movement between the display and the keyboard.

COMMENT: Experienced users might key coded menu selections in a standard area identified only by its consistent location and use. If the system is designed primarily for novice users, however, that entry area should be given an appropriate label, such as `ENTER choice here: __`

REFERENCE: MS 5.15.4.2.2; PR 4.6.3.

SEE ALSO: 3.1.5•2, 4.0•6.

•9 **Feedback for Menu Selection**

When a user has selected and entered a control option from a menu, if there is no immediately observable natural response then the computer should display some other acknowledgment of that entry.

COMMENT: An explicit message might be provided. In some applications, however, it may suffice simply to highlight the selected option label (e.g., by brightening or inverse video) when that would provide an unambiguous acknowledgment.

REFERENCE: MS 5.15.4.1.12.

SEE ALSO: 1.1•5, 3.0•14, 4.2•1, 4.2•10.

•10 **Explanatory Title for Menu**

Display an explanatory title for each menu, reflecting the nature of the choice to be made.

EXAMPLE: (Good)
```
Organizational Role
  r = Responsible
  a = Assigned
  p = Performing
```

(Bad)
```
Select:
  r = Responsible
  a = Assigned
  p = Performing
```

REFERENCE: BB 1.9.1.

Menu Options Worded as Commands •11

The wording of menu options should consistently represent commands to the computer, rather than questions to the user.

EXAMPLE: For option selection by pointing, a "+" (or some other special symbol) might be used consistently to distinguish a selectable control option from other displayed items, as

(Good) `+PRINT`

(Bad) `PRINT?`

EXAMPLE: For option selection by code entry, the code for each option should be consistently indicated, as

(Good) `p = Print`

(Bad) `Print? (Y/N)`

COMMENT: Wording options as commands will permit logical selection by pointing, will facilitate the design of mnemonic codes for keyed entry, and will help users learn commands in systems where commands can be used to by-pass menus.

COMMENT: Wording options as commands implies properly that the initiative in sequence control lies with the user. Wording options as questions implies initiative by the computer.

REFERENCE: PR 4.6.8.

SEE ALSO: 3.1.3•20, 4.0•23.

▶ Option Wording Consistent with Command Language •12

If menu selection is used in conjunction with or as an alternative to command language, design the wording and syntactic organization of displayed menu options to correspond consistently to defined elements and structure of the command language.

COMMENT: Where appropriate, display cumulative sequences of menu selections in a command entry area until the user signals entry of a completely composed command.

COMMENT: This practice will speed the transition for a novice user, relying initially on sequential menu selection, to become an experienced user composing coherent commands without such aid.

REFERENCE: MS 5.15.4.2.9; Badre, 1984.

SEE ALSO: 3.1.3•35, 4.0•15.

3.1.3 Dialogue Type - Menu Selection

•13 Letter Codes for Menu Selection

If menu selections are made by keyed codes, design each code to be the initial letter or letters of the displayed option label, rather than assigning arbitrary letter or number codes.

EXAMPLE: (Good)
```
m = Male
f = Female
```
(Bad)
```
1 = Male
2 = Female
```

EXCEPTION: Options might be numbered when a logical order or sequence is implied.

EXCEPTION: When menu selection is from a long list, the line numbers in the list might be an acceptable alternative to letter codes.

COMMENT: Several significant advantages can be cited for mnemonic letter codes. Letters are easier than numbers for touch-typists to key. It is easier to memorize meaningful names than numbers, and thus letter codes can facilitate a potential transition from menu selection to command language when those two dialogue types are used together. When menus have to be redesigned, which sometimes happens, lettered options can be reordered without changing codes, whereas numbered options might have to be changed and so confuse users who have already learned the previous numbering.

COMMENT: Interface designers should **not** create unnatural option labels just to ensure that the initial letter of each will be different. There must be some natural differences among option names, and special two- or three-letter codes can probably be devised as needed to emphasize those differences. In this regard, there is probably no harm in mixing single-letter codes with special multiletter codes in one menu.

REFERENCE: BB 1.3.6; MS 5.15.4.2.11; Palme, 1979; Shinar, Stern, Bubis and Ingram, 1985.

SEE ALSO: 4.0•13.

▶ Consistent Coding of Menu Options •14

If letter codes are used for menu selection, use those letters consistently in designating options from one transaction to another.

EXAMPLE: As a negative example, the same action should **not** be given different names (and hence different codes) at different places in a transaction sequence, such as

(Bad) f = Forward and n = Next

EXAMPLE: As a negative example, the same code should **not** be given to different actions, such as

(Bad) q = Quit and q = Queue

COMMENT: Different codes for the same action will tend to confuse users and impede learning. The same code for different actions will tend to induce user errors, especially if those actions are frequently taken. However, this practice may be tolerable when selections are seldom taken, and then always taken from labeled alternatives.

SEE ALSO: 3.1.3•19, 4.0•15.

▶ Standard Symbol for Prompting Entry •15

Choose a standard symbol for indicating that an entry is required, and reserve that symbol only for that purpose.

EXAMPLE: (Good) ENTER organization type:

(Bad) ENTER organization type

COMMENT: Some standard prompting symbol, such as the colon shown in the example here, will help to cue users that an input is required. The same symbol should be used to prompt data entries, code entries for menu selections, command entries, etc.

REFERENCE: BB 2.5.2.

SEE ALSO: 1.4•9, 4.4•10.

Explicit Option Display •16

When control entries for any particular transaction will be selected from a small set of options, show those options in a menu added to the working display, rather than requiring a user to remember them or to access a separate menu display.

COMMENT: A complete display of control options will sometimes leave little room for display of data. If an extensive menu must be added to a working data display, provide that menu as a separate window that can temporarily overlay displayed data at user request, but can then be omitted again by further user action.

REFERENCE: MS 5.15.4.1.5.

SEE ALSO: 4.4•5, and Section 2.7.5.

3.1.3 Dialogue Type - Menu Selection

•17 ▶ **Complete Display of Menu Options**

Design a menu to display all options appropriate to any particular transaction.

EXCEPTION: A familiar set of general control options (i.e., options that are **always** implicitly available) may be omitted from individual displays; such general options might be selected by requesting a general menu, or perhaps by function key or command entry.

SEE ALSO: 4.4•1, and Section 3.2.

•18 ▶ **Menu Options Dependent on Context**

Design a menu to display only those options that are actually available in the current context for a particular user.

EXAMPLE: Privileged users might be shown more options than regular users.

EXAMPLE: Displayed file directories should contain only those files actually available to the particular user.

EXAMPLE: Offer a CHANGE option only to users who are authorized to make changes to the particular data being displayed.

EXCEPTION: Menu displays for a system under development might display future options not yet implemented, but such options should be specially marked in some way so that users will understand that they are not available.

COMMENT: If a user selects a displayed option, and is then told that option is not actually available, an undesirable element of unpredictability has been introduced into the interface design. Users may become uncertain and confused about sequence control. Also irritated.

REFERENCE: BB 1.8.11; MS 5.15.4.2.3.

SEE ALSO: 3.2•10, 4.4•1.

•19 **Consistent Display of Menu Options**

When menus are provided in different displays, design them so that option lists are consistent in wording and ordering.

EXAMPLE: As a negative example, if + PRINT is the last option in one menu, the same print option should **not** be worded + COPY at the beginning of another menu.

REFERENCE: MS 5.15.4.2.4.

SEE ALSO: 3.1.3•14, 4.0•15.

Menus Distinct from Other Displayed Information •20

If menu options are included in a display that is intended also for data review and/or data entry, which is often a practical design approach, ensure that they are distinct from other displayed information; locate menu options consistently in the display and incorporate some consistent distinguishing feature to indicate their special function.

EXAMPLE: All control options might be displayed beginning with a special symbol, such as a plus sign (+ NEXT, + BACK, etc.).

COMMENT: An interesting variation in menu design is the use of "embedded menus" in which various items within a working display are highlighted in some way to indicate that they can be selected to obtain further information. Thus a text display of encyclopedia information might highlight certain words as cross references to related material, words which can be selected in context rather than from some separate menu listing. Here the selectable items are made visually distinct without being segregated spatially.

REFERENCE: Koved and Shneiderman, 1986.

SEE ALSO: 2.3•8, 3.1.8•5, 4.0•8.

Logical Ordering of Menu Options •21

List displayed menu options in a logical order; if no logical structure is apparent, then display the options in order of their expected frequency of use, with the most frequent listed first.

EXAMPLE: (Good)
```
i = Initiate track
m = Move track
d = Delete track
```
(Bad)
```
d = Delete track
i = Initiate track
m = Move track
```

EXAMPLE: [See sample displays at the end of this section.]

REFERENCE: BB 2.9.4; EG 2.3.1; MS 5.15.4.2.5; PR 4.6.6; Palme, 1979.

SEE ALSO: 2.3•2, 2.5•17.

3.1.3 Dialogue Type - Menu Selection

•22 **Logical Grouping of Menu Options**

Format a menu to indicate logically related groups of options, rather than as an undifferentiated string of alternatives.

EXAMPLE: In vertical listing of options, subordinate categories might be indented.

EXAMPLE: [See sample displays at the end of this section.]

COMMENT: Logical grouping of menu options will help users learn system capabilities.

COMMENT: When logical grouping requires a trade-off against expected frequency of use, interface designers should resolve that trade-off consistently for those functions throughout the menu structure.

REFERENCE: EG 2.2.8, 2.3; Foley and Wallace, 1974; Liebelt, McDonald, Stone and Karat, 1982; McDonald, Stone and Liebelt, 1983.

SEE ALSO: 4.4•3.

•23 ▶ **Logical Ordering of Grouped Options**

If menu options are grouped in logical subunits, display those groups in a logical order; if no logical structure is apparent, then display the groups in the order of their expected frequency of use.

EXAMPLE: [See sample displays at the end of this section.]

REFERENCE: PR 4.6.6.

•24 ▶ **Labeling Grouped Options**

If menu options are grouped in logical subunits, give each group a descriptive label that is distinctive in format from the option labels themselves.

EXAMPLE: [See sample displays at the end of this section.]

COMMENT: Although this practice might sometimes seem to waste display space, it will help provide user guidance. Moreover, careful selection of group labels may serve to reduce the number of words needed for individual option labels.

REFERENCE: MS 5.15.3.1.10.

SEE ALSO: 4.4•4.

Hierarchic Menus for Sequential Selection •25

When menu selection must be made from a long list, and not all options can be displayed at once, provide a hierarchic sequence of menu selections rather than one long multipage menu.

EXAMPLE: [See sample displays at the end of this section.]

EXCEPTION: Where a long list is already structured for other purposes, such as a list of customers, a parts inventory, a file directory, etc., it might be reasonable to require the user to scan multiple display pages to find a particular item. Even in such cases, however, an imposed structure for sequential access may prove more efficient, as when a user can make preliminary letter choices to access a long alphabetic list.

COMMENT: Beginning users may learn faster and understand better a menu permitting a single choice from all available options, when those can be displayed on one page. However, a single long menu that extends for more than one page will hinder learning and use. The interface designer can usually devise some means of logical segmentation to permit several sequential selections among few alternatives instead of a single difficult selection among many.

REFERENCE: MS 5.15.4.2.6; Dray, Ogden and Vestewig, 1981.

SEE ALSO: 4.4•4.

▶ General Menu •26

Provide a general menu of basic options as the top level in a hierarchic menu structure, a "home base" to which a user can always return as a consistent starting point for control entries.

COMMENT: Return to the general menu might be accomplished by an OPTIONS function key, or by an explicitly labeled option on every display, or by a generally available implicit option.

SEE ALSO: 3.1.3•34, 3.2•2.

3.1.3 Dialogue Type - Menu Selection

•27 ▶ **Minimal Steps in Sequential Menu Selection**

When users must step through a sequence of menus to make a selection, design the hierarchic menu structure to minimize the number of steps required.

EXAMPLE: [See sample displays at the end of this section.]

COMMENT: This represents a trade-off against the need for logical grouping in hierarchic menus. Minimize the number of hierarchic levels, but not at the expense of display crowding.

REFERENCE: MS 5.15.4.1.6; Miller, 1981; Snowberry, Parkinson, and Sisson, 1983.

•28 ▶ **Easy Selection of Important Options**

When hierarchic menus are used, design their structure to permit immediate user access to critical or frequently selected options.

EXAMPLE: [See sample displays at the end of this section.]

COMMENT: It may be desirable in general purpose systems whose use is varied and unpredictable, to permit users to tailor menu design (particularly the general menu) to their individual needs, so that the options used most frequently will appear first for each user.

COMMENT: In designing fixed hierarchic menus, if frequent or critical options do appear logically at lower levels, and so will be less accessible, some design alternatives should be considered. For a critical action, some sort of "panic" option might be included in **every** menu display, or might be implemented by function key. For frequent actions, some special menu display might be provided as a supplementary shortcut to the designed menu hierarchy.

REFERENCE: MS 5.15.4.2.8.

SEE ALSO: 3.1.4•2.

▶ Automatic Cursor Placement •29

On separate menu displays (i.e., for menus not included with data displays), when menu selection is by pointing the computer should place the cursor automatically at the first listed option; when menu selection is by code entry, place the cursor in the command entry area.

COMMENT: When menu selection is by code entry, for some applications it may increase the efficiency of sequence control if a null entry is recognized as a default to the first displayed option (assuming that the first option is the most likely choice). If that is done, it should be done consistently.

SEE ALSO: 1.4•28.

Indicating Current Position in Menu Structure •30

When hierarchic menus are used, display to the user some indication of current position in the menu structure.

EXAMPLE: [See sample displays at the end of this section.]

COMMENT: One possible approach would be to recapitulate prior (higher) menu selections on the display. If routine display of path information seems to clutter menu formats, then a map of the menu structure might be provided at user request as a HELP display.

REFERENCE: MS 5.15.4.1.6; Billingsley, 1982.

SEE ALSO: 4.4•4.

▶ Control Options Distinct from Menu Branching •31

Format the display of hierarchic menus so that options which actually accomplish control entries can be distinguished from options which merely branch to other menu frames.

EXAMPLE: [See sample displays at the end of this section.]

COMMENT: In some applications, it may prove efficient to design "hybrid" menus which display one branch of the menu hierarchy elaborated to include all of its control options while other branches are simply indicated by summary labels. In such a hybrid menu, it will help orient users if options that accomplish control actions are highlighted in some way to distinguish them from options which will result in display of other frames of the hierarchic menu.

3.1.3 Dialogue Type - Menu Selection

•32

▶ Consistent Design of Hierarchic Menus

When hierarchic menus are used, ensure that display format and selection logic are consistent at every level.

REFERENCE: MS 5.15.4.1.6.

SEE ALSO: 4.0•6.

•33

▶ Return to Higher-Level Menus

When hierarchic menus are used, require users to take only one simple key action to return to the next higher level.

COMMENT: This action could be considered analogous to the BACKUP option proposed as an interrupt for sequence control.

REFERENCE: BB 4.4.4.

SEE ALSO: 3.3•4.

•34

▶ Return to General Menu

When hierarchic menus are used, require users to take only one simple key action to return to the general menu at the top level.

EXAMPLE: [See sample displays at the end of this section.]

COMMENT: This action could be considered analogous to the REVIEW option proposed as an interrupt for sequence control.

SEE ALSO: 3.1.3•26, 3.3•5.

By-Passing Menu Selection with Command Entry •35

Allow experienced users to by-pass a series of menu selections and make an equivalent command entry directly.

COMMENT: In effect, a command entry might specify an option anywhere in a hierarchic menu structure, permitting a user to jump down several levels, or to move directly from one branch to another.

COMMENT: If a command by-passes only a portion of the complete menu sequence, and so does not yet specify a complete control entry, then display the appropriate next menu to guide completion of the control entry.

REFERENCE: BB 2.8, 4.5; PR 4.7.3; Badre, 1984.

SEE ALSO: 3.0•2, 3.1.3•12, 4.4•31.

▶ Stacking Menu Selections •36

For menu selection by code entry, when a series of selections can be anticipated before the menus are displayed, permit a user to combine those selections into a single "stacked" entry.

COMMENT: If necessary, stacked sequential entries might be separated by some character, such as a space, slash, comma or semicolon. It would be preferable, however, if they were simply strung together without special punctuation. Computer interpretation of an unpunctuated string will require letter codes (by preference) or fixed-digit number codes for option selection.

REFERENCE: BB 2.9; Badre, 1984.

SEE ALSO: 3.1.3•13, 3.2•13 thru •17, 3.5•4, 3.5•5.

3.1.3 Menu Selection

(Good) **Sample Menu Display** **(Good)**

```
W : WORD PROCESSING MENU                GO = General
                                             Options
    D : DOCUMENT MANAGEMENT
        C    =  Create
                CF  = Free format
                CL  = Letter
                CM  = Memo
                CW  = Wide format

        E    =  Edit
        P    =  Print
        CO   =  COpy
        RE   =  REname
        DE   =  DElete
        SP   =  SPelling check
        I    =  Index

    T  = Transferring documents
    L  = List processing
    S  = Status information
    U  = User profile


ENTER letter code to select action or another menu.
█
```

These sample displays represent portions of a large menu of word processing functions. The good menu indicates the current position in a hierarchic menu structure. Different levels in the hierarchic structure are indicated by indentation. This menu offers (bolded) control actions for the most frequently used branch ("document management"), along with options to select other branches in the menu hierarchy. Selection of another branch would show a similar menu display, offering control actions within the selected branch, but without offering the control actions shown here for document management.

The bad menu shows an alternative design for the same functions. The bad menu lacks hierarchic structure, and does not distinguish between control actions and options that merely select further menus. The bad menu would require several successive menu selections in order to take frequent actions.

(Bad) **Sample Menu Display** **(Bad)**

```
C   =  Create a new document

CW  =  Create a new wide document

D   =  Delete a document

E   =  Edit an existing document

F   =  Finished — Exit

I   =  Index of documents

L   =  List Processing

M   =  More Menu selections

P   =  Print a document

S   =  Spelling Error Detection



Type the letters followed by a RETURN
█
```

This bad menu display violates in some degree several design guidelines in this section:

- 3.1.3•21 Logical ordering of menu options
- •22 Logical grouping of menu options
- •23 Logical ordering of grouped options
- •24 Labeling grouped options
- •25 Hierarchic menus for sequential selection
- •27 Minimal steps in sequential menu selection
- •28 Easy selection of important options
- •30 Indicating current position in menu structure
- •31 Control options distinct from menu branching
- •34 Return to general menu

3.1.4 Dialogue Type - Function Keys

Function keys permit control entries by direct selection of labeled keys, rather than from displayed menus.

•1 **Function Keys for Critical Control Entries**

Consider function keys for tasks requiring only a limited number of control entries, or for use in conjunction with other dialogue types as a ready means of accomplishing critical entries that must be made quickly without syntax error.

REFERENCE: BB 4.4; MS 5.15.2.3.1, 5.15.4.4.

•2 **▶ Function Keys for Frequent Control Entries**

Consider function keys for frequently required control entries.

EXAMPLE: Commonly used function keys include ENTER, PRINT, NEXT PAGE, PREV PAGE, OPTIONS, etc.

COMMENT: When frequently used options are always available via function keys, they need not be included in displayed menus.

REFERENCE: BB 4.4; MS 5.15.2.3.1.

SEE ALSO: 3.1.3•28, and Section 3.2.

•3 **▶ Function Keys for Interim Control Entries**

Consider function keys for interim control entries, i.e., for control actions taken before the completion of a transaction.

EXAMPLE: Function keys will aid such interim actions as DITTO, CONFIRM, and requests for PRINT, or HELP, and also interrupts such as BACKUP, CANCEL, etc.

COMMENT: Interim control refers to an action taken by a user while working with displayed data, e.g., while still keying data entries or changes, etc. Function keys will aid interim control entries partly because those entries may be frequent. More importantly, however, function keys permit those control entries to be made without special cursor positioning, so that they do not interfere with data entry.

Distinctive Labeling of Function Keys •4

Label each function key informatively to designate the function it performs; make labels sufficiently different from one another to prevent user confusion.

EXAMPLE: As a negative example of uninformative labeling, cited from an actual design, logging onto a system should **not** be initiated by a key labeled PANIC.

EXAMPLE: As a negative example of confusingly similar labeling, two keys should **not** be labeled ON and DN.

REFERENCE: BB 4.4.7; MS 5.15.2.3.9, 5.15.3.1.10.c.

SEE ALSO: 4.0•10.

▶ Labeling Multifunction Keys •5

If a key is used for more than one function, always indicate to the user which function is currently available.

COMMENT: If a key is used for just two functions, depending upon defined operational mode, then alternate illuminated labels might be provided on the key to indicate which function is current. In those circumstances, it is preferable that only the currently available function is visible, so that the labels on a group of keys will show what **can** be done at any point.

COMMENT: If key function is specific to a particular transaction, provide an appropriate guidance message on the user's display to indicate the current function.

REFERENCE: MS 5.15.2.4.2, 5.15.2.4.4.

SEE ALSO: 4.4•13.

Single Keying for Frequent Functions •6

Keys controlling frequently used functions should permit single key action and should **not** require double (control/shift) keying.

▶ Logical Pairing of Double-Keyed Functions •7

If double (control/shift) keying is used, the functions paired on one key should be logically related.

EXAMPLE: If a particular function key moves the cursor to the upper left corner of a display screen, then that same key when shifted might be used to move the cursor to the bottom right corner of the screen.

EXAMPLE: As a negative example, a function key that moves the cursor should **not** be used when shifted to delete displayed data.

REFERENCE: Stewart, 1980.

3.1.4 Dialogue Type - Function Keys

•8 ▶ **Consistent Logic for Double Keying**

If double (control/shift) keying is used, the logical relation between shifted and unshifted functions should be consistent from one key to another.

EXAMPLE: One consistent logic might be that shifted and unshifted functions are opposite, so that if a particular key moves the cursor forward then that key when shifted would move the cursor backward.

EXAMPLE: Another possible logic might be that shifted and unshifted functions are related by degree, so that if a particular key deletes a single displayed character then that key when shifted would delete a word.

COMMENT: Consistency in the underlying logic for double keying will help a user to learn the functions associated with different keys.

•9 **Single Activation of Function Keys**

Ensure that any key will perform its labeled function with a single activation, and will **not** change its function with repeated activation.

EXCEPTION: On a very compact keypad, where separate keys are not available to accommodate the range of needed functions, it might be acceptable to group logically related functions on a single key, where repeated key activation would extend the range of control action in a consistent way, e.g., to DELETE character, word, sentence, or paragraph with repeated keystrokes.

REFERENCE: MS 5.15.2.3.7.

SEE ALSO: 4.0•2.

•10 **Feedback for Function Key Activation**

When function key activation does not result in any immediately observable natural response, provide users with some other form of computer acknowledgment.

COMMENT: Temporary illumination of the function key will suffice, if key illumination is not used for other purposes such as indicating available options. Otherwise an advisory message should be displayed.

COMMENT: As an interesting variation, user guidance **prior** to key activation might be provided, where partial depression of a double-contact function key would explain its use, either by voice output ("talking keyboard") or by visual display of a HELP message.

REFERENCE: MS 5.15.2.3.8; Geiser, Schumacher and Berger, 1982.

SEE ALSO: 3.0•14, 4.2•1.

Indicating Active Function Keys •11

If some function keys are active and some are not, indicate the current subset of active keys in some noticeable way, perhaps by brighter illumination.

COMMENT: This practice will speed user selection of function keys.

REFERENCE: MS 5.15.2.4.3; Hollingsworth and Dray, 1981.

SEE ALSO: 4.4•13.

▶ Disabling Unneeded Function Keys •12

When function keys are not needed for any current transaction, temporarily disable those keys under computer control; do **not** require users to apply mechanical overlays for this purpose.

COMMENT: If a user selects a function key that is invalid for the current transaction, no action should result except display of an advisory message indicating what functions **are** available at that point.

REFERENCE: MS 5.15.9.1; PR 4.12.4.5.

SEE ALSO: 3.2•10, 3.5•1, 6.0•11.

Single Key for Continuous Functions •13

When a function is continuously available, assign that function to a single key.

REFERENCE: MS 5.15.2.3.4.

Consistent Assignment of Function Keys •14

If a function is assigned to a particular key in one transaction, assign that function to the same key in other transactions.

EXAMPLE: A SAVE key should perform the same function at any point in a transaction sequence.

COMMENT: This becomes a design issue, of course, only in applications where the set of needed functions does vary somewhat from one transaction to another.

REFERENCE: BB 4.4.8; MS 5.15.2.3.2; Foley and Wallace, 1974.

3.1.4 Dialogue Type - Function Keys

•15 **▶ Consistent Functions in Different Operational Modes**

When a function key performs different functions in different operational modes, assign equivalent or similar functions to the same key.

EXAMPLE: A particular key might be used to confirm data changes in one mode, confirm message transmission in another, etc.

EXAMPLE: As a negative example, a key labeled RESET should **not** be used to save data in one mode, dump data in another, and signal task completion in a third (cited from an actual design).

REFERENCE: Stewart, 1980.

•16 **Easy Return to Base-Level Functions**

If the functions assigned to a set of keys change as a result of user selection, give the user an easy means to return to the initial, base-level functions.

EXAMPLE: In cockpit design, where multifunction keys may be used for various purposes such as navigation or weapons control, the pilot should be able to take a single action to restore those keys quickly to their basic flight control functions.

COMMENT: In effect, multifunction keys can provide hierarchic levels of options much like menu selection dialogues, with the same need for rapid return to the highest-level menu.

COMMENT: For some applications, it may be desirable to automate the return to base-level assignment of multifunction keys, to occur immediately on completion of a transaction and/or by time-out following a period of user inaction. The optimum period for any automatic time-out would have to be determined empirically for each application.

REFERENCE: Aretz and Kopala, 1981.

•17 **Distinctive Location**

Group function keys in distinctive locations on the keyboard to facilitate their learning and use; place frequently used function keys in the most convenient locations.

REFERENCE: MS 5.15.2.3.6.

SEE ALSO: 4.0•8.

•18 **▶ Layout Compatible with Use**

The layout of function keys should be compatible with their importance; give keys for emergency functions a prominent position and distinctive coding (e.g., size and/or color); provide physical protection for keys with potentially disruptive consequences.

SEE ALSO: 6.0•12.

Command language permits a user to specify desired control actions by composing messages to a computer.

Command Language •1

Consider command-language dialogues for tasks involving a wide range of control entries, where users may be highly trained and will use the system frequently.

COMMENT: Command language should also be considered for tasks where control entries may be mixed with data entries in arbitrary sequence, such as when making flight reservations. Such applications will generally require extensive user training.

REFERENCE: MS 5.15.4.5.1; Martin, 1973.

Standard Display Area for Command Entry •2

When command language is used for sequence control, provide a command entry area in a consistent location on every display, preferably at the bottom.

COMMENT: Adjacent to the command entry area there should be a display window reserved for prompting entries, for recapitulation of command sequences (with scrolling to permit extended review), and to mediate question-and-answer dialogue sequences (i.e., prompts and responses to prompts).

REFERENCE: EG 2.3; MS 5.15.4.5.7; Granda, Teitelbaum and Dunlap, 1982.

SEE ALSO: 2.5•11, 3.1.3•8, 4.0•6.

Functional Wording •3

Design a command language so that a user can enter commands in terms of functions desired, without concern for internal computer data processing, storage and retrieval mechanisms.

EXAMPLE: Users should be able to request display of a data file by name alone, without any further specification such as that file's location in computer storage.

COMMENT: Where file names are not unique identifiers, the computer should be programmed to determine whatever further context is necessary for identification. Or perhaps the computer should ask the user to designate a "directory" defining the subset of files of current interest.

REFERENCE: MS 5.15.4.1.10.

3.1.5 Dialogue Type - Command Language

•4 **Layered Command Language**

Design a command language so that its functions are organized in groups (or "layers") for ease in learning and use.

EXAMPLE: A user should be able to display the next of a set of received messages with some simple command such as READ NEXT, although a complete command to retrieve **any** message might include potential specification of which message, from which message list, in which format, to which output device.

COMMENT: The fundamental layer of the language should be the easiest, allowing use of the system by people with little training and/or limited needs. Successive layers of the command language can then increase in complexity for users with greater skills. In effect, simple versions of commands can be recognized by defaulting all of the optional parameters.

COMMENT: Control forms might be used to display default options for complicated commands.

REFERENCE: Reisner, 1977.

SEE ALSO: 3.1.2•3, 4.4•31.

•5 **Meaningful Command Names**

Choose command names that are meaningful, and that specifically describe the functions being implemented.

COMMENT: In some systems, functions are arbitrarily assigned letters as command names, e.g., the letter D preceded by a special key such as CONTROL might be a LOG-OFF command. In such cases, when command names are not real words that describe system functions, users will have difficulty learning to use the system.

COMMENT: If users are permitted to enter abbreviations rather than complete command names, ensure that users are told the command name represented by the abbreviation. Otherwise, a short abbreviation may seem an arbitrary code. For instance, a prompt might read

    To DELETE a record, enter D

rather than

    To erase a record, enter D.

REFERENCE: Grudin and Barnard, 1984.

▶ Familiar Wording •6

Choose words for a command language that reflect the user's point of view, and correspond to the user's operational language.

EXAMPLE: To transfer a file, the assigned command should be something like TRANSFER, MOVE, or SEND, and **not** some jargon term like PIP.

REFERENCE: EG 4.1.1, 4.2.12, 4.2.13; MS 5.15.4.5.2.

SEE ALSO: 4.0•16, 4.0•17.

▶ Consistent Wording of Commands •7

Design all words in a command language, and their abbreviations, to be consistent in meaning from one transaction to another, and from one task to another.

EXAMPLE: As a negative example, do **not** use EDIT in one place, MODIFY in another, UPDATE in a third, all referring to the same kind of action.

EXAMPLE: Choose wording so that commands will be congruent with one another, following natural language patterns; if one command is UP, its complement should be DOWN; other natural complements include RIGHT-LEFT, FORWARD-BACK, IN-OUT, PUSH-PULL, RAISE-LOWER, etc.

REFERENCE: EG 4.2.9, 4.2.13; MS 5.15.4.5.6; Carroll, 1982; Demers, 1981.

SEE ALSO: 4.0•15, 4.0•17.

▶ Distinctive Meaning for Commands •8

Design words in a command language so that they are distinctive from one another, and emphasize significant differences in function.

COMMENT: In general, do **not** give different commands semantically similar names, such as SUM and COUNT, or ERASE and DELETE, or QUIT and EXIT. System design abounds with negative examples of similarly named commands which confuse their users: DISPLAY and VIEW (where one command permits editing displayed material and one does not), COMPOSE and CREATE (where one command sends a composed message to an outbox and one leaves a message on the desk), etc. Even experienced users will make errors with such commands.

COMMENT: Some researchers deal with this question by recommending the use of specific rather than general command names.

REFERENCE: BB 3.7.5; MS 5.15.4.5.3; Barnard, Hammond, MacLean and Morton, 1982.

3.1.5 Dialogue Type - Command Language

•9 ▶ **Distinctive Spelling for Commands**

Design words and abbreviations in a command language with distinctive spelling, so that simple spelling errors will be recognized as such rather than invoking a different command.

EXAMPLE: As a negative example, if one command name is DELETE, abbreviated DEL, then another command should **not** be named DELIVER, with an abbreviation of DELR. Instead, ERASE could be substituted for DELETE, or SEND for DELIVER.

COMMENT: When a system has only a few commands, all of those commands should be distinctive. When a system has many commands, it may not be possible to ensure that each is distinctive. In that case, it is important to ensure that any commands which are destructive or time-consuming are made distinctive.

REFERENCE: BB 2.2.5.

•10 **User-Assigned Command Names**

Design a command language with flexibility to permit a user to assign personal names to files, frequently used commands, etc.

COMMENT: Frequently used commands should be easy for a user to enter. Where users differ in the frequency of the commands they use, perhaps the designer should provide for flexibility in command naming. On the other hand, users will **not** be perfectly consistent in specifying command names, and a carefully designed set of commands might well prove better for some applications.

COMMENT: For users who must move back and forth between different systems with differently defined command languages, some flexibility in command naming might permit those users to establish their own consistent terminology.

COMMENT: Before users can be allowed to adopt their own assigned command names, the computer must check those names to prevent duplication.

COMMENT: A potential risk of increased flexibility is increased confusion, if users forget what names they have specified for commands and data files. The computer should maintain a current index of command and file names for on-line user reference.

REFERENCE: Carroll, 1982.

SEE ALSO: 3.2•18, 4.4•18, 4.4•19.

User-Requested Prompts •11

Allow users to request computer-generated prompts as necessary to determine required parameters in a command entry, or to determine available options for an appropriate next command.

EXAMPLE: Using a HELP function key, or perhaps simply keying a question mark in the command entry area, would be satisfactory methods to request prompting.

COMMENT: In some applications it may be desirable to let an inexperienced user simply choose a general "prompted mode" of operation, where any command entry produces automatic prompting of (required or optional) parameters and/or succeeding entry options.

REFERENCE: MS 5.15.4.5.8; Demers, 1981.

SEE ALSO: 4.4•7, 4.4•12.

▶ General List of Commands •12

Provide a general list of basic commands, with appropriate command format guidance, that will always be available to serve as a "home base" or consistent starting point for composing command entries.

COMMENT: Such a general list of commands might provide more comprehensive user guidance than is possible when prompting command entry from a working display.

SEE ALSO: 3.2•2, 4.4•2.

Command Stacking •13

Allow users to key a series of commands at one time ("command stacking").

COMMENT: This practice will allow experienced users to by-pass prompting sequences. Command stacking will reduce the extended memory load on users. Command stacking may also be much faster than separate entry of commands, in systems where input/output processing is overloaded by multiple users.

REFERENCE: BB 2.9.

SEE ALSO: 3.2•13, 4.4•12.

3.1.5 Dialogue Type - Command Language

•14 ▶ **User Definition of Macro Commands**

Allow users to assign a single name to a defined series of commands and then use that named "macro" for subsequent command entry.

COMMENT: In this way users can make a frequent but complicated task easier to accomplish, when the interface designer has failed to anticipate that particular need.

REFERENCE: Demers, 1981; Foley and Wallace, 1974.

SEE ALSO: 3.2•18.

•15 **Minimal Punctuation**

Allow users to enter commands without any punctuation other than the spaces between words.

COMMENT: Command entry will be faster and more accurate when spaces are used rather than any other kind of punctuation.

REFERENCE: MS 5.15.4.5.4; Radin, 1984.

SEE ALSO: 3.2•16.

•16 ▶ **Standard Delimiter**

If command punctuation other than spaces is required, perhaps as a delimiter to distinguish optional parameters or to separate entries in a stacked command, adopt a single standard delimiter symbol for that purpose.

EXAMPLE: A slash (/) might be a good choice.

COMMENT: Whatever symbol is adopted as a delimiter for command entries should preferably be the same as any delimiter that might be used when making data entries.

COMMENT: Note, however, that even if some single delimiter is specified for consistent use in command punctuation, command entry will be slower and less accurate than if no delimiter at all were required. People do not punctuate reliably.

SEE ALSO: 1.4•4, 3.2•17.

•17 **Ignoring Blanks in Command Entry**

Treat single and multiple blanks between words as equivalent when processing command entries.

COMMENT: People cannot readily distinguish one blank space from several, and so the computer should not impose such a distinction.

SEE ALSO: 1.0•30.

Abbreviation of Commands •18

Allow users to abbreviate commands.

EXAMPLE: If a "P" uniquely identifies a print command (i.e., no other commands start with "P") then a user should be able to enter PRINT, or PR, or P, or any other truncation to initiate printing.

COMMENT: As a corollary, misspelled command entries should also be tolerated, within the limits of computer recognition. The computer can interrogate a user as necessary to resolve ambiguous entries.

COMMENT: If a command language is still changing, as during system development, do **not** permit variable abbreviation. For the user, an abbreviation that works one day may not work the next. For the software designer, the addition of any new command might require revision of recognition logic for other commands.

REFERENCE: BB 2.4.3; Demers, 1981.

Standard Techniques for Command Editing •19

Allow users to edit erroneous commands with the same techniques that are employed to edit data entries.

COMMENT: Consistent editing techniques will speed learning and reduce errors.

Interpreting Misspelled Commands •20

Where the set of potential command entries is well defined, program the computer to recognize and execute common misspellings of commands, rather than requiring re-entry.

COMMENT: This practice should permit a sizable reduction in wasted keying without serious risk of misinterpretation. The necessary software logic is akin to that for recognizing command abbreviations.

COMMENT: For novice users, it may be helpful for the computer to display an inferred command for user confirmation before execution.

REFERENCE: BB 2.2.4; MS 5.15.7.10; Gade, Fields, Maisano, Marshall and Alderman, 1981.

3.1.5 Dialogue Type - Command Language

•21 ▶ **Recognizing Command Synonyms**

If a system will have many novice or infrequent users, ensure that the computer can recognize a variety of synonyms for each word defined in the command language.

EXAMPLE: The words "mail", "post", and "transmit" might be accepted synonyms for the command "send".

COMMENT: What synonyms are frequently employed can be determined by analysis of user error records in prototype testing.

COMMENT: Infrequent users may need to relearn command names each time they use the system. For those users, time spent learning commands is not worthwhile considering that they will seldom use those commands.

COMMENT: Command synonyms will be helpful for users who are experienced with different systems. For instance, if users of different editors must occasionally use the same mail system, that mail system might permit synonyms for such common functions as creating and storing documents. If user experience with other systems is known, as when all users of a mail system use one of two editors, designers should include appropriate synonyms. Otherwise, the users themselves might be permitted to tailor command names to employ familiar terminology.

COMMENT: If most system users will gain expertise through frequent use, then documentation and error messages should be provided to help new users learn accepted commands, rather than permitting them to enter command synonyms. When a command language has been carefully designed, with easily distinguishable command names and rule-based abbreviations, frequent users will benefit from time spent learning that command language rather than designing their own language.

REFERENCE: Good, Whiteside, Wixon and Jones, 1984.

▶ Recognizing Alternative Syntax •22

If a system will have many novice or infrequent users, ensure that the computer can recognize probable alternative forms of command syntax.

EXAMPLE: The computer might accept alternative methods of specifying a document, such as "memo 3", "memo #3", "#3", or simply "3"; users might be allowed to use different punctuation and/or to list command modifiers in different orders.

COMMENT: What alternative syntax should be recognized can be determined by analysis of user error records in prototype testing. Recognition of alternative syntax will require more complex parsing of commands, perhaps enlarging by several times that segment of interface software. But that effort will be justified by increased recognition of user entries.

COMMENT: Infrequent users may need to relearn syntax rules each time they use the system. For those users, time spent learning syntax is not worthwhile considering that they will seldom use that syntax.

COMMENT: Recognizing alternative syntax will be helpful for users who are experienced with different systems. For instance, if users of different editors must occasionally use the same mail system, that mail system might permit alternative syntax corresponding to the syntax of those different editors. If user experience with other systems is known, as when all users of a mail system use one of two editors, designers should provide for those different syntax forms. Otherwise, the users themselves might be permitted to tailor command syntax to employ familiar forms.

COMMENT: If most system users will gain expertise through frequent use, then documentation and error messages should be provided to help new users learn accepted syntax, rather than permitting them to use alternative syntax forms. When a command language has been carefully designed to minimize keystrokes and errors, and ensure that similar commands require the same syntax, frequent users will benefit from time spent learning acceptable syntax rather than designing their own language.

REFERENCE: Good, Whiteside, Wixon and Jones, 1984.

Correcting Command Entry Errors •23

If a command entry is not recognized, allow the user to revise the command rather than rejecting the command outright.

COMMENT: Misstated commands should not simply be rejected. Instead, software logic should guide users toward proper command formulation. Preserve the faulty command for reference and modification, and do **not** require a user to rekey the entire command just to change one part.

SEE ALSO: 3.5•2, 4.3•15.

3.1.5 Dialogue Type - Command Language

•24 ▶ **Replacing Erroneous Commands**

If a user makes a command entry error, after the error message has been displayed allow the user to enter a new command; a user should **not** be forced to correct and complete an erroneous command.

COMMENT: In considering a command entry error message, a user may decide that the wrong command was chosen in the first place, and wish to substitute another command instead.

•25 **Reviewing Destructive Commands**

If a command entry may have disruptive consequences, require the user to review and confirm a displayed interpretation of the command before it is executed.

SEE ALSO: 3.5•8, 6.0•18.

Query language is a special form of command language that can be used to request information from a computer.

Query Language •1

Consider query language dialogue for tasks emphasizing unpredictable information retrieval (as in many analysis and planning tasks), with moderately trained users.

COMMENT: Guidelines for command language design would apply equally to query languages.

REFERENCE: Ehrenreich, 1981.

SEE ALSO: Section 3.1.5.

Natural Organization of Data •2

Design a query language so that it reflects a data structure or organization perceived by users to be natural.

EXAMPLE: If a user supposes that all data about a particular person are stored in one place, then the query language should probably permit such data to be retrieved by a single query, even though actual computer storage might carry the various data in different files.

COMMENT: The users' natural perception of data organization can be discovered by survey or experimentation. When users' perceptions do not match the actual structure of computer-stored data, then special care will be needed to preserve the users' viewpoint in query language design.

REFERENCE: Durding, Becker and Gould, 1977.

▶ Coherent Representation of Data Organization •3

Establish one single representation of the data organization for use in query formulation, rather than multiple representations.

EXAMPLE: If different queries will access different data bases over different routes, a user should not necessarily need to know this.

COMMENT: Beginners or infrequent users may be confused by different representational models.

REFERENCE: Michard, 1982.

SEE ALSO: 4.4•18.

3.1.6 Dialogue Type - Query Language

•4 Task-Oriented Wording

Design a query language so that the wording of a query simply specifies what data are requested; a user should **not** have to tell the computer how to find the data.

COMMENT: This objective has been called "nonprocedurality", meaning that a user should not have to understand computer procedures for finding data.

REFERENCE: Michard, 1982.

•5 Flexible Query Formulation

Allow users to employ alterative forms when composing queries, corresponding to common alternatives in natural language.

EXAMPLE: When quantifying a query, a user should be able to employ equivalent forms, such as "over 50", "more than 50", "51 or more".

REFERENCE: Michard, 1982.

•6 Minimal Need for Quantifiers

Design a query language to minimize the need for quantifiers in query formulation.

EXAMPLE: Negative quantifiers ("no", "none", "zero", etc.) are particularly difficult for users to deal with; other potentially confusing quantifiers include indefinite ("some", "any") and interrogative ("how many") forms.

COMMENT: People have difficulty in using quantifiers. If a query language does require quantifiers, it may be helpful to allow a user to select the desired quantifier from a set of sample queries worded to maximize their distinctiveness.

Logic to Link Queries •7

Design a query language to include logic elements that permit users to link sequential queries as a single entry.

EXAMPLE: Common links for query formulation include "and", "or", etc.

COMMENT: However a query language should be designed so that it does not **require** logical links. Some logical quantifiers ("greater than", "less than", etc.) may confuse users. As an alternative to logical linking, it may prove helpful to allow a user to formulate a series of simple queries to narrow the set of retrieved data.

COMMENT: It may help a user to specify logical links accurately if the computer can display a graphical depiction of relations among data sets as those relations are specified during query composition. One researcher recommends Venn diagrams for this purpose.

REFERENCE: Michard, 1982.

SEE ALSO: 3.1.8•7.

Linking Sequential Queries •8

Design a query language to allow the logical linking of sequential queries.

EXAMPLE: Logical linking of queries might be accomplished with referential pronouns ("of them", "of those") that will be recognized by the computer in terms of current context.

Confirming Large-Scale Retrieval •9

If a query will result in a large-scale data retrieval, require the user to confirm the transaction or else take further action to narrow the query before processing.

COMMENT: In this regard, it may be helpful to permit a user to set some upper bound for data output, in effect to define what constitutes a "large-scale" retrieval.

COMMENT: It may help a user to decide whether to confirm or modify a pending query, if the user can request a **partial** display of the currently specified data output.

3.1.7 Dialogue Type - Natural Language

> *Natural language recognition might permit a novice user to compose commands without any special training.*

•1 Constrained Natural Language

Consider using some constrained form of natural language dialogue in applications where task requirements are broad ranging and poorly defined, and where little user training can be provided.

COMMENT: Computer processing of natural language is now being developed on an experimental basis. Current capabilities permit computer recognition of constrained forms of "natural" language, with some limits on vocabulary and syntax. Such constrained natural languages might be considered akin to command languages, with the drawback that they are probably not as carefully designed.

COMMENT: For untrained users, the seemingly familiar form of a (constrained) natural language dialogue may help introduce them to computer capabilities. Such users may manage to do **something** right from scratch, without having to surmount an initial hurdle of learning more specialized command languages and control procedures. As users gain experience, they may eventually learn more efficient methods of interacting with the system. On the other hand, infrequent computer users may forget whatever training they receive, and so remain novices indefinitely.

COMMENT: Do **not** consider using unconstrained natural language dialogues for current interface design. Even if a computer can be programmed to recognize unconstrained natural language, it is not clear whether that would help in any defined information handling task. A natural language will often cause confusion in communication among its **human** users. Something better may be needed to mediate human communication with computers. For applications where task requirements are well defined, other types of dialogue will probably prove more efficient.

REFERENCE: Hayes, 1985; Shneiderman, 1981.

Graphic interaction permits a user to select displayed control elements by pointing and other direct manipulation.

Control of Graphic Data •1

For users who must work with graphic data, provide control capabilities as recommended in guidelines for graphic data entry.

COMMENT: Methods of user interaction with graphic data may be so complex that they can be exploited fully only by skilled users. Design recommendations for that specialized kind of graphic interaction are discussed in guidelines pertaining to graphic data entry (Section 1.6).

SEE ALSO: Section 1.6.

Graphic Control Aids for Casual Users •2

For casual users, consider providing graphic aids to supplement other types of sequence control.

COMMENT: Advocates recommend simple graphic interaction techniques as an aid for casual users, so that they will not have to learn more complicated methods of sequence control such as command entry. It is that potential use of graphic interaction to simplify control dialogue that is discussed in this section of the guidelines.

3.1.8 Dialogue Type - Graphic Interaction

•3 **Iconic Menus**

When system users have different linguistic backgrounds, consider providing graphic menus which display icons to represent the control options.

EXAMPLE: A computer-based information system at an international airport might display graphic menus with icons to indicate simple control options.

COMMENT: Here "icon" is intended to mean a small graphic figure which represents a control operation or object.

COMMENT: Some advocates recommend the use of icons whenever possible in place of verbal labels or explanations. They argue that icons can have universal meaning for users with different linguistic backgrounds. Whereas verbal labels may require translation into other languages for different user groups.

COMMENT: Some critics, however, are concerned that the meaning of icons may **not** be clear. Careful testing may be required to develop a satisfactory set of icons in terms of both legibility and interpretability. And even then it may prove a wise precaution to supplement icons by displaying redundant verbal labels.

COMMENT: One serious drawback of iconic menus is that they will **not** permit the sequential concatenation of coded menu selections that can ease the transition to command entry as novice users become more experienced. Thus iconic menus are more appropriate for casual rather than continuing use.

REFERENCE: Barnard and Marcel, 1984; Bewley, Roberts, Schroit and Verplank, 1983; Foley and Van Dam, 1982; Muter and Mayson, 1986; Smith, Irby, Kimball and Verplank, 1982.

SEE ALSO: 2.4•13.

•4 **▶ Supplementary Verbal Labels**

If icons are used to represent control actions in menus, display a verbal label with each icon to help assure that its intended meaning will be understood.

COMMENT: Some of the objects and processes dealt with in sequence control are necessarily abstract, and so may be difficult to depict by iconic representation. A redundant verbal label might help make the meaning clear to a user who is uncertain just what a displayed icon means.

COMMENT: One skeptic of iconic representation has cited the problems of early logographic languages, such as Egyptian hieroglyphs, and reminds us, "It took about 2500 years to get rid of the iconic shapes that we are now reviving for computer workstations."

REFERENCE: Bigelow, 1985.

Direct Manipulation •5

For casual system users, consider providing a capability for direct manipulation of displayed objects as a means of sequence control.

EXAMPLE: Rather than compose a command or select a function key to file a document, a user might move a displayed icon representing the document to superimpose it on another icon representing a file.

COMMENT: In sequence control by direct manipulation, the techniques for selecting and moving displayed objects would be similar to those described in guidelines for graphic data entry.

COMMENT: An extension of this idea is the use of "embedded menus" in which various items within a working display are highlighted in some way to indicate that they can be selected to obtain further information. Thus a text display of encyclopedia information might highlight certain words as cross references to related material, words which can be selected in context rather than from some separate menu listing.

COMMENT: Advocates recommend direct manipulation as a means of enhancing a user's understanding of control actions, arguing that such manipulation can help clarify the relations among abstract objects and processes. Others recommend manipulation as a simple alternative to learning a command language, arguing that it is easier for a user to see and point than to remember and type.

COMMENT: Critics argue that for experienced users direct manipulation will generally not be as efficient as other methods of sequence control. If direct manipulation is provided, some other more efficient alternative such as command language should also be available for those users who can learn it. Unfortunately, direct manipulation suffers the same drawback as that cited for iconic menus, namely, this mode of graphic interaction does **not** aid transition to the use of command language as novice users gain experience.

REFERENCE: Koved and Shneiderman, 1986; Shneiderman, 1982.

SEE ALSO: Section 1.6.

3.1.8 Dialogue Type - Graphic Interaction

•6 **Graphic Display of Control Context**

Consider graphic means for displaying to users the context of current control actions.

EXAMPLE: A graphic representation of the currently selected values of functions, elements and attributes affecting control actions might help reduce user errors in sequence control.

EXAMPLE: Graphic techniques might be used to display the scope of a proposed control action, such as outlining a passage of text (or other group of display elements) currently selected for deletion.

SEE ALSO: 1.3•7, 1.6•7, 1.6•12, 4.4•13, and Section 3.4.

•7 **Graphic Display of Control Prompting**

Consider graphic means for displaying to users prompting aids and other guidance pertaining to current control actions.

EXAMPLE: A guidance display providing a graphic representation of keypad layout with notes explaining the various key functions might help a novice user to learn the control options available via function keys.

EXAMPLE: A graphic representation of command syntax might aid language learning by novice users who could be confused by other forms of symbolic notation.

EXAMPLE: A graphic representation of logical combinations specified in query formulation might help reduce errors in the use of query language.

REFERENCE: Bauer and Eddy, 1986; Michard, 1982; Shneiderman, 1982.

SEE ALSO: 3.1.6•7.

Transaction selection refers to the control actions and computer logic that initiate transactions.

User Control in Transaction Selection •1

Allow users to select transactions; computer processing constraints should **not** dictate sequence control.

EXAMPLE: A user who wants to interrupt a current activity should **not** be required by the computer to complete some long sequence of useless transactions.

COMMENT: When a logical sequence of transactions can be determined in advance, interface design might encourage and help a user to follow that sequence. Guidance may be desirable though constraint is not.

REFERENCE: PR 4.6.7.

SEE ALSO: 3.0•1, 3.0•4, 3.0•5, 4.0•2.

General List of Control Options •2

Provide a general list of basic control options that will always be available to serve as a "home base" or consistent starting point for control entries.

COMMENT: Return to this starting point can be accomplished by an OPTIONS function key, or by an explicit control option on every display, or by a generally available implicit option.

COMMENT: Such a capability may be helpful even when all dialogue is user-initiated. It might be the general menu for a menu selection dialogue, or might be a standard starting point for composing command entries.

COMMENT: However a user should not be **required** to return to a display of general options in order to make a control entry. If a user remembers option codes or commands, ideally those control entries could be made from any point in a transaction sequence.

REFERENCE: BB 4.1; PR 3.3.16.

SEE ALSO: 3.1.3•26, 3.1.5•12, 4.4•2.

▸ Organization and Labeling of Listed Options •3

Design the general options list to show control entry options grouped, labeled and ordered in terms of their logical function, frequency and criticality of use, following the general guidelines for menu design.

SEE ALSO: 4.4•2, 4.4•3, and Section 3.1.3.

•4 **Indicating Appropriate Control Options**

Make available to users a list of the control options that are specifically appropriate for any transaction.

COMMENT: Transaction-specific options might be listed in the working display if there is space for them. Otherwise, they might be displayed in an overlay window at user request.

COMMENT: Treat control options that are available for almost any transaction as **implicit** options, which need not be included in a list of transaction-specific options unless they are particularly appropriate to the current transaction. One convenient way to offer implicit options is via function keys, although some experienced users may prefer to select implicit options by command entry.

SEE ALSO: 3.1.4•2, 4.4•1, 4.4•6.

•5 **▶ Prompting Control Entries**

Provide users with whatever information may be needed to guide control entries at any point in a transaction sequence, by incorporating prompts in a display and/or by providing prompts in response to requests for HELP.

REFERENCE: MS 5.15.4.1.4.

SEE ALSO: 4.4•1.

•6 **Cursor Placement for Pointing at Options**

When users will select among displayed options by pointing, place the cursor on the first (most likely) option at display generation.

SEE ALSO: 3.1.3•29, 4.4•16.

•7 **▶ Cursor Placement for Keyed Entry of Options**

When users must select options by keyed entry of a corresponding code, place the cursor in the control entry area at display generation.

REFERENCE: PR 4.7.1.

SEE ALSO: 4.4•16.

•8 **Displaying Option Codes**

When users must select options by code entry, display the code associated with each option in a consistent distinctive manner.

EXAMPLE: In many applications an equal sign can be used to designate option codes, such as

`N = Next page, P = Prev page,` etc.

Task-Oriented Wording for Options •9

Employ task-oriented wording for control options to reflect the user's view of the current transaction.

EXAMPLE: When assigning aircraft to a mission, the relevant control option should be ASSIGN rather than ENTER.

SEE ALSO: 4.0•17.

Only Available Options Offered •10

Offer users only control options that are actually available for the current transaction.

COMMENT: If certain options are not yet implemented, as during system development, or are not available for any other reason, those should be annotated on the display.

SEE ALSO: 3.1.3•18, 3.1.4•12, 6.0•11.

Indicating Control Defaults •11

When control is accomplished by keyed command or option code entries, if a default is defined for a null control entry then indicate that default to the user.

EXAMPLE: `Press ENTER to see more options.`

EXCEPTION: If a consistent default is adopted throughout interface design, that default need not be explicitly indicated for each individual transaction.

COMMENT: Here the phrase "null control entry" refers to pressing an ENTER key without first keying a command or option code (and without any accompanying data). It does not refer to defaults for optional parameters that might accompany a valid control entry, whose values might be displayed only at user request.

COMMENT: It is not necessary that **any** defaults be defined for null control entries. In such cases, the computer might simply respond

`ENTER alone is not recognized here.`

The point here is that when defaults **are** defined, and when they vary from one transaction to another, then users should be informed of the current default logic.

SEE ALSO: 4.4•7.

•12 **Consistent CONTINUE Option**

At any step in a defined transaction sequence, if there is only a single appropriate next step then provide a consistent control option to continue to the next transaction.

EXAMPLE: CONTINUE or NEXT or STEP might be suitable names for this option.

EXCEPTION: If data entry is involved, then require a user to take an explicit ENTER action to signal data entry, rather than simply selecting CONTINUE.

REFERENCE: PR 4.11.

SEE ALSO: 4.4•5.

•13 **Stacked Control Entries**

Allow users to key a sequence of commands or option codes as a single "stacked" control entry.

COMMENT: In particular, allow users to enter stacked entries from any menu so that an experienced user can make any specific control entry without having to view subsequent menus.

COMMENT: Control entry stacking may be helpful when a user is being prompted to enter a series of parameter values, and knows what several succeeding prompts will request and what values to enter.

COMMENT: Control entry stacking will permit a transition from simple step-by-step control entry by novice users, as in menu selection and question-and-answer dialogues, to the entry of extended command-language statements by experienced users; entry stacking is especially helpful in time-shared systems where computer response to any user entry may be slow.

REFERENCE: EG 6.2, 6.2.1; PR 2.6, 4.7.3; Palme, 1979.

SEE ALSO: 3.1.3•36, 3.1.5•13, 3.5•4, 3.5•5.

•14 ▸ **Consistent Order in Entry Stacking**

For control entry stacking, require entries to be in the same order as they would normally be made in a succession of separate control entry actions.

REFERENCE: EG 6.2.1.

▶ Abbreviation in Entry Stacking •15

For control entry stacking, accept command names or their abbreviations or option codes just as if those control entries had been made separately.

COMMENT: In some applications, it might prove helpful if the computer were to display its interpretation of a stacked entry for user review and confirmation.

REFERENCE: EG 6.2.1.

▶ Minimal Punctuation of Stacked Entries •16

Allow users to stack control entries without any punctuation other than spaces between words or option codes.

COMMENT: Sometimes stacked entries may require specific symbols as delimiters for their interpretation. Careful design of command languages and/or option codes can minimize the need for delimiters to interpret correct entries. Delimiters may still be needed, however, to protect against possible user errors, i.e., stacked commands that have been improperly composed.

COMMENT: Entry will be faster and more accurate when spaces are used rather than any other kind of punctuation.

REFERENCE: Radin, 1984.

SEE ALSO: 3.1.5•15.

▶ Standard Delimiter in Entry Stacking •17

If punctuation other than spaces is needed to separate entries in a stacked control entry, adopt a single standard symbol for that purpose.

EXAMPLE: A slash (/) may be a good choice.

COMMENT: Whatever symbol is adopted as a delimiter for control entries should preferably be the same as any delimiter that might be used when making data entries.

COMMENT: Note that even when a standard symbol is consistently used to punctuate stacked entries, entry will be slower and less accurate than if only spaces are used for punctuation.

REFERENCE: EG 6.2.1.

SEE ALSO: 1.4•4, 3.1.5•16.

•18 **User Definition of Macro Commands**

Provide flexibility in transaction selection by allowing users to assign a single name to a defined series of control entries, and then use that named "macro" for subsequent command entry.

COMMENT: In this way users can make frequently required but complicated tasks easier to accomplish, when the interface designer has failed to anticipate a particular need.

REFERENCE: Demers, 1981; Foley and Wallace, 1974.

SEE ALSO: 3.1.5•10, 3.1.5•14.

•19 **User-Specified Transaction Timing**

When appropriate to task requirements, allow users to specify transaction timing, i.e., when a requested transaction should start or should be completed, or the periodic scheduling of repeated transactions.

EXAMPLE: A user might wish to specify that a requested data analysis routine be deferred until some later hour, to ensure that interim updates to the data will be taken into account.

EXAMPLE: A user might prepare a number of messages for transmittal, but specify that actual transmission be deferred until a later time.

EXAMPLE: A user might wish to specify that a request for a large printout be deferred to take advantage of reduced overnight rates, but specify a printout deadline to ensure delivery by 0800 the next morning.

EXAMPLE: A user might wish to specify that summarized briefing material be prepared automatically at 0600 every morning until further notice.

COMMENT: In many applications, of course, users will wish specified transactions performed as quickly as possible. In some applications, however, users may have good reasons to delay initiation (or completion) of transactions.

COMMENT: In some instances, it may be possible to provide appropriate software logic to aid user decisions on transaction timing. For example, if a user requested a bulky printout, the computer might propose overnight printing as an economical alternative, subject to user confirmation.

COMMENT: Allowing users to specify periodic scheduling for routine transactions will tend to reduce the memory load on users, and may help ensure more reliable system performance. If such routine transactions require further user inputs, as when preparing periodic activity reports, computer logic can be devised to prompt users on a timely basis to make the necessary entries.

Interrupt capabilities that permit a user to change ongoing transactions allow flexibility in sequence control.

User Interruption of Transactions •1

Provide flexibility in sequence control by allowing a user to interrupt or cancel a current transaction, in ways appropriate to task requirements.

COMMENT: Provision of flexible interrupt capabilities will generally require some sort of suspense file or other buffering in software design. Some such capability, however, will be needed for other reasons, e.g., to allow users to correct mistaken entries, and to permit the computer to require user confirmation of potentially destructive entries.

REFERENCE: PR 3.3.16, 3.3.17.

Distinctive Interrupt Options •2

If different kinds of user interrupt are provided, design each interrupt function as a separate control option with a distinct name.

EXAMPLE: As a negative example, it would **not** be good design practice to provide a single INTERRUPT key which has different effects depending upon whether it is pushed once or twice; users would be confused by such an expedient and uncertain about what action has been taken and its consequences.

CANCEL Option •3

If appropriate to sequence control, provide a CANCEL option which will have the effect of erasing any changes just made by the user and restoring the current display to its previous version.

EXAMPLE: In a sequence of related data entries, on several display frames, CANCEL might erase ("clear") data in the current frame as a convenient way to begin keying corrected data, rather than having to erase each data item individually.

COMMENT: The easiest way to implement CANCEL may be simply to regenerate the current display.

REFERENCE: Foley and Wallace, 1974.

SEE ALSO: 1.4•2.

•4 **BACKUP Option**

If appropriate to sequence control, provide a nondestructive BACKUP option which will have the effect of returning to the display for the last previous transaction.

EXAMPLE: In a sequence of related data entries, on several display frames, BACKUP might return to the previous frame, where data items could then be erased by CANCEL or could be edited individually.

COMMENT: Such a BACKUP capability will prove feasible only in the software design of well-defined transaction sequences, but will prove helpful when it **can** be provided.

COMMENT: In some applications, BACKUP might be designed to include cancellation of any interim entries made in a pending transaction. More often, however, it will be better to preserve pending entries without processing. Interface design should be consistent in that regard.

REFERENCE: MS 5.15.7.7; Foley and Van Dam, 1982; Foley and Wallace, 1974.

SEE ALSO: 1.4•2.

•5 **REVIEW Option**

If appropriate to sequence control, provide a nondestructive REVIEW option which will have the effect of returning to the first display in a defined transaction sequence, permitting the user to review a sequence of entries and make necessary changes.

EXAMPLE: In a sequence of related data entries, on several display frames, REVIEW might return to the first frame, from which data could be reviewed and edited as needed throughout the sequence of frames.

COMMENT: REVIEW is an extension of the BACKUP capability, and is useful only in well-defined transaction sequences such as step-by-step data entry in a question-and-answer dialogue.

COMMENT: In some applications, REVIEW might be designed to include cancellation of any interim entries made in a pending transaction. More often, however, it will be better to preserve pending entries without processing. Interface design should be consistent in that regard.

SEE ALSO: 1.4•2.

RESTART Option •6

If appropriate to sequence control, provide a RESTART option which will have the effect of canceling any entries that may have been made in a defined transaction sequence and returning to the beginning of the sequence; when data entries or changes will be nullified by a RESTART action, require users to CONFIRM the RESTART.

EXAMPLE: In a sequence of related data entries, on several display frames, RESTART might erase all data entries in the sequence and return to the first frame.

COMMENT: A RESTART action combines the functions of REVIEW and CANCEL, and is relevant only to well-defined transaction sequences.

REFERENCE: BB 4.7; MS 5.15.7.5.d.

SEE ALSO: 3.0•21, 4.2•11, 6.0•5.

END Option •7

If appropriate to sequence control, provide an END option which will have the effect of concluding a repetitive transaction sequence.

EXAMPLE: In a repetitive sequence of data entries, where completing one transaction cycles automatically to begin the next, END might break the cycle and permit the user to select other transactions.

COMMENT: END can be implemented by whatever means are appropriate to the dialogue design, i.e., by menu selection, command entry, or function key.

REFERENCE: EG 4.2.10.

PAUSE and CONTINUE Options •8

If appropriate to sequence control, provide PAUSE and CONTINUE options which will have the effect of interrupting and later resuming a transaction sequence without any change to data entries or control logic for the interrupted transaction.

EXAMPLE: A user might wish to interrupt a current task to read an incoming message.

EXAMPLE: In the interests of data protection, as a "security pause", a user might wish to blank a current display to prevent its being read by some casual visitor.

COMMENT: A "security pause" may have to be implemented quickly and easily, which suggests that this option should be offered via function key.

SEE ALSO: 6.2•6.

•9 ▸ **Indicating PAUSE Status**

If a PAUSE option is provided, display some indication of the PAUSE status whenever that option is selected by a user, and prompt the CONTINUE action that will permit resumption of the interrupted transaction.

•10 **SUSPEND Option**

If appropriate to sequence control, provide a SUSPEND option which will have the effect of preserving current transaction status when a user leaves the system, and permitting resumption of work at that point when the user later logs back onto the system.

COMMENT: In the interests of data protection, a SUSPEND option might require special user identification procedures at subsequent log-on, to prevent unauthorized access to suspended transactions.

SEE ALSO: 6.1•8.

•11 ▸ **Indicating SUSPEND Status**

If a SUSPEND option is provided, display some indication of the SUSPEND status whenever that option is selected by a user, and at subsequent log-on prompt the user in those procedures that will permit resumption of the suspended transaction.

Context definition by the computer will help ensure that control actions are related to a user's current task.

Defining Context for Users •1

Design the sequence control software to maintain context for the user throughout the series of transactions comprising a task; where appropriate, display the results of previous entries affecting present actions, and indicate currently available options.

SEE ALSO: 1.8•9, 3.0•9, 4.4•13.

▶ Context Established by Prior Entries •2

Design the sequence control software to interpret current control actions in the context of previous entries; do **not** require users to re-enter data.

EXAMPLE: If data have just been stored in a named file, permit users to request a printout of that file without having to re-enter its name.

EXCEPTION: If transactions involving contextual interpretation would have destructive effects (e.g., data deletion), then display the interpreted command first for user confirmation.

COMMENT: The software logic supporting contextual interpretation of control entries need not be perfect in order to be helpful. The computer may occasionally have to present an interpreted command for user review and approval. That is still easier for the user than having to specify every command completely in the first place.

REFERENCE: MS 5.15.2.1.6; PR 2.3.

3.4 Context Definition

•3 ▶ **Record of Prior Entries**

Permit users to request a summary of the results of prior entries to help determine present status.

EXAMPLE: In an aircraft assignment task, there might be a status display showing current commitments of aircraft to missions.

EXAMPLE: In a text processing application, there might be a status display listing documents already edited and printed in the current work session.

COMMENT: Summarizing prior entries will be particularly helpful in tasks where transaction sequences are variable, where a user must know what was done in order to decide what to do next. Summarizing prior entries may not be needed for routine transactions if each step identifies its predecessors explicitly, although even in those circumstances a user may be distracted and at least momentarily become confused.

REFERENCE: EG 4.2.7.

SEE ALSO: 3.1.1•3, 4.4•22.

•4 **Display of Operational Mode**

When context for sequence control is established in terms of a defined operational mode, remind users of the current mode and other pertinent information.

EXAMPLE: If text is displayed in an editing mode, then a caption might indicate EDIT as well as the name of the displayed text; if a DELETE mode is selected for text editing, then some further displayed signal should be provided.

REFERENCE: BB 4.3.4; EG 4.2.1; MS 5.15.5.5, 5.15.7.5.a.

SEE ALSO: 4.1•5, 4.2•8, 4.4•13.

Display of Control Parameters •5

Allow users to review any control parameter(s) that are currently operative.

EXAMPLE: A text processing system might display a variety of parameters to control document printing, including margin widths, line spacing, number of copies, etc., which would represent current default values that a user could review and change when desired.

EXAMPLE: A system might display a "user profile" that specifies for a particular user which editor will be used, which message system, which general menu, what level of prompting, etc.

COMMENT: A capability for parameter review is helpful even when a user selects all parameters personally. Users will sometimes be forgetful, or may become confused, particularly if their activities are interrupted for any reason.

REFERENCE: MS 5.15.4.1.5.

SEE ALSO: 4.2•8, 4.4•13.

Highlighting Selected Data •6

When a user is performing an operation on some selected display item, highlight that item.

COMMENT: This practice will help avoid error, if a user has misunderstood or perhaps forgotten which item was selected.

REFERENCE: EG 2.1.1.

SEE ALSO: 4.2•10.

Consistent Display of Context Information •7

Ensure that information displayed to provide context for sequence control is distinctive in location and format, and consistently displayed from one transaction to the next.

REFERENCE: MS 5.15.4.4.

SEE ALSO: 4.0•6.

3.5 Error Management

Error management by the computer will help prevent user errors and correct those errors that do occur.

•1 **Appropriate Response to All Entries**

Design the interface software to deal appropriately with all possible control entries, correct and incorrect.

EXAMPLE: If a user selects a function key that is invalid for a particular transaction, no action should result except display of an advisory message indicating what functions are appropriate at that point.

COMMENT: For certain routine and easily recognized errors, such as trying to tab beyond the end of a line, a simple auditory signal ("beep") may be sufficient computer response.

REFERENCE: PR 4.12.4.5.

SEE ALSO: 3.1.4•12, 6.0•14.

•2 **Command Editing**

Allow users to edit an extended command during its composition, by backspacing and rekeying, before taking an explicit action to ENTER the command.

COMMENT: Users can often recognize errors in keyed entries prior to final entry.

REFERENCE: EG 5.4; MS 5.15.7.2; Neal and Emmons, 1984.

SEE ALSO: 1.4•2, 3.1.5•23, 6.0•10, 6.3•8.

•3 **Prompting Command Correction**

If an element of a command entry is not recognized, or logically inappropriate, prompt users to correct that element rather than requiring re-entry of the entire command.

EXAMPLE: A faulty command can be retained in the command entry area of the display, with the cursor automatically positioned at the incorrect item, plus an advisory message describing the problem.

REFERENCE: BB 5.2.1; EG 4.2.2, 4.2.3; MS 5.15.7.1.

SEE ALSO: 4.3•15.

Errors in Stacked Commands •4

If an error is detected in a stacked series of command entries, either consistently execute to the point of error, or else consistently require users to correct errors before executing **any** command.

COMMENT: It may help the user if the commands are executed to the point of error, or it may not. In most applications, partial execution will probably prove desirable. The point here is that a considered interface design decision should be made and then followed consistently.

REFERENCE: EG 5.6; PR 4.7.3.

▶ Partial Execution of Stacked Commands •5

If only a portion of a stacked command can be executed, notify the user and provide appropriate guidance to permit correction, completion, or cancelation of the stacked command.

COMMENT: Note that stacked commands can fail because of error in their composition, or for other reasons such as unavailability of required data.

REFERENCE: MS 5.15.7.11; Dwyer, 1981.

SEE ALSO: 4.4•7.

Explicit Entry of Corrections •6

When a user completes correction of an error, whether of a command entry or data entry, require the user to take an explicit action to re-enter the corrected material; use the same ENTER action for re-entry that was used for the original entry.

REFERENCE: MS 5.15.7.9; PR 4.12.4.6.

SEE ALSO: 1.0•9, 3.0•5, 6.0•9, 6.3•11.

User Confirmation of Destructive Entries •7

When a control entry will cause any extensive change in stored data, procedures and/or system operation, and particularly if that change cannot be easily reversed, notify the user and require confirmation of the action before implementing it.

REFERENCE: BB 5.6; EG 4.1.2, 4.2.8; MS 5.15.7.4, 5.15.7.5.b; Foley and Wallace, 1974.

SEE ALSO: 4.3•18, 6.0•18, 6.0•20.

•8 ▶ **User Warned of Potential Data Loss**

Word the prompt for a CONFIRM action to warn users explicitly of any possible data loss.

EXAMPLE:
(Good)
CONFIRM deletion of entire AIRFIELD file??
(Bad)
CONFIRM DELETE

SEE ALSO: 3.1.5•25, 4.3•17, 6.0•17.

•9 ▶ **Distinctive CONFIRM Action**

Provide an explicitly labeled CONFIRM function key, different from the ENTER key, for user confirmation of questionable control and data entries.

COMMENT: Confirmation should **not** be accomplished by pushing some other key **twice.**

COMMENT: Some interface designers recommend that in special cases confirmation should be made more difficult still, e.g., by keying the separate letters C-O-N-F-I-R-M. Even such extreme measures, however, cannot guarantee that users will not make errors.

SEE ALSO: 3.1.4•3, 3.1.4•4, 3.1.4•9, 3.1.4•14, 6.0•19.

•10 **UNDO to Reverse Control Actions**

Ensure that any user action can be immediately reversed by an UNDO command.

EXAMPLE: If a user is overhasty in confirming a destructive action, and realizes the mistake right away (i.e., before taking another action), then an UNDO action might be taken to reverse the damage.

COMMENT: UNDO itself should be reversible, so that a second UNDO action will do again whatever was just undone.

COMMENT: Such an UNDO capability is currently available in many interface designs, and should be provided more generally. Even with an UNDO capability, however, a user may make an irretrievable mistake, if succeeding actions intervene before a prior destructive action is noticed.

REFERENCE: Lee and Lochovsky, 1983; Nickerson and Pew, 1971; Shneiderman, 1982.

SEE ALSO: 6.0•21.

▶ Preventing Data Loss at LOG-OFF •11

When a user requests LOG-OFF, check pending transactions and if any pending transaction will not be completed, or if data will be lost, display an advisory message requesting user confirmation.

EXAMPLE:

```
Current data entries have not been filed;
SAVE if needed, before confirming LOG-OFF.
```

COMMENT: A user may sometimes suppose that a job is done before taking necessary implementing actions.

REFERENCE: BB 4.8; MS 5.15.7.5.e.

SEE ALSO: 4.3•17, 6.3•22.

▶ Immediate Data Correction •12

If a data entry transaction has been completed and errors detected, permit users to make corrections directly and immediately.

COMMENT: It is helpful to correct data entry errors at the source, i.e., while a user still has the entry in mind and/or source documents at hand. When a user cannot correct an entry, as when transcribing from a source document that itself contains an error, it may help to allow the user to defer entry of the wrong item. Alternatively, the user might wish to cancel the transaction.

COMMENT: For transactions involving extended entry of multiple items, computer checking might be invoked as each page or section of data is entered.

REFERENCE: EG 5.7; PR 2.5.

SEE ALSO: 1.7•6, 6.3•9.

▶ Flexible BACKUP for Error Correction •13

Allow users to BACKUP easily to previous steps in a transaction sequence in order to correct an error or make any other desired change.

REFERENCE: MS 5.15.7.7.

SEE ALSO: 3.3•4, 6.3•12.

3.6 Alarms

Alarms and alerting signals generated by the computer might be controlled by users in terms of logic and operation.

•1 **Alarm Definition by Users**

In monitoring and process control applications, allow users to define the conditions (in terms of variables and values) that will result in computer generation of alarm messages.

EXAMPLE: The nurse in charge of an intensive care monitoring station might need to specify for each patient that a warning be signaled when blood pressure (a "variable") exceeds or falls below defined levels ("values").

EXCEPTION: There are some situations where alarm conditions must be predefined by functional, procedural, or perhaps even legal requirements, such as violation of aircraft separation in air traffic control.

SEE ALSO: 4.1•10.

•2 **Distinctive and Consistent Alarms**

Ensure that alarm signals and messages are distinctive for each class of events.

COMMENT: Users should participate in the classification of alarming events, and might help in specifying the desired nature of different alarm signals.

SEE ALSO: 4.3•19.

•3 **Alarm Acknowledgment**

Provide users with a simple means of acknowledging and turning off non-critical alarm signals.

EXAMPLE: A function key labeled ALARM OFF would suffice for this purpose.

▶ Alarm Reset •4

Provide users with a simple means of turning off an auditory alarm, without erasing any displayed message that accompanies the auditory signal.

EXAMPLE: A function key labeled ALARM RESET would suffice for this purpose.

COMMENT: Turning off an auditory alarm will ensure that any succeeding alarm condition will be able to generate a **new** auditory signal to attract the user's attention.

▶ Special Acknowledgment of Critical Alarms •5

If users are required to acknowledge a special or critical alarm in some special way, ensure that such acknowledgment will not inhibit or slow remedial user response to the critical initiating condition.

COMMENT: Do not make acknowledgment of critical alarms too complicated. Help users deal with the **cause** of an alarm rather than the symptom.

3.7 Design Change

Design change of software supporting control functions may be needed to meet changing operational requirements.

•1 **Flexible Design for Sequence Control**

When sequence control requirements may change, which is often the case, provide some means for the user (or a system administrator) to make necessary changes to control functions.

COMMENT: Sequence control functions that may need to be changed include those represented in these guidelines, namely, the types of dialogue that are provided, procedures for transaction selection and interrupt, methods for context definition and error management, and alarm control.

SEE ALSO: 3.0•1, 3.1.3•28, 3.1.5•10, 3.1.5•14, 3.2•18, 3.2•19, 3.6•1.

Section 4

USER GUIDANCE

USER GUIDANCE

User guidance refers to error messages, alarms, prompts, and labels, as well as to more formal instructional material provided to help guide a user's interaction with a computer. The fundamental objectives of user guidance are to promote efficient system use (i.e., quick and accurate use of full capabilities), with minimal memory load on the user and hence minimal time required to learn system use, and with flexibility for supporting users of different skill levels.

User guidance should be regarded as a pervasive and integral part of interface design that contributes significantly to effective system operation. User guidance cannot be merely a decoration added at the end, like frosting on a cake. A study by Magers (1983) has demonstrated convincingly that good user guidance can result in faster task performance, fewer errors, greater user satisfaction, and will permit accomplishment of information handling tasks otherwise impossible for novice users.

A narrow view of user guidance deals with preventing and correcting user errors. But minimizing user errors may require improvements in broad aspects of interface design — in techniques for data display, in procedures for data entry and sequence control — as well as provision of user guidance. Moreover, any general consideration of user guidance functions must include provision of status information, job aids, and routine feedback, as well as feedback for error correction.

Many user interface design features contribute directly or indirectly to guide a user's interaction with an on-line computer system. The primary principle governing this aspect of interface design is to maintain consistency. With consistent interface design, users can learn to apply computer tools more quickly, more accurately, and with more confidence.

Design consistency implies predictability of system response to user inputs. A fundamental principle is that **some** response be received. For every action (input) by the user there should be a noticeable reaction (output) from the computer. It is this feedback linking action to reaction that defines each discrete transaction and maintains user orientation in interaction with the system.

The importance of feedback information for guiding the user has been emphasized by Engel and Granda in their recommendations for user interface design:

> Feedback to user action covers keeping the user informed of where he is, what he has done, and whether it was successful. . . . In an interactive computing situation, immediate feedback by the system is important in establishing the user's confidence and satisfaction with the system. One of the more frustrating aspects of any interactive system is sitting at the terminal after entering something and waiting for a response. Questions arise such as, 'Is the system still going?', 'Is the terminal still connected to the system?', 'Did the computer lose my input?', 'Is the system in a loop?'. A message that indicates that the system is still working on the problem or a signal that appears while the system is processing the user's input provides the user with the necessary assurance that everything is all right.
>
> (1975, page 13)

Introduction

Predictability of computer response is related to system response time. Timely response can be critical in maintaining user orientation to the task. If a response is received only after a long delay, the user's attention may have wandered. Indeed, the user may forget just which action the machine is responding **to**. Frequent user actions, generally those involving simple inputs such as function key or menu selections, should be acknowledged immediately. In transactions where output must be deferred pending the results of computer search and/or calculation, the expected delay should be indicated to the user in a quick interim message.

Some experts argue that consistency of system response time may be more important in preserving user orientation than the absolute value of the delay, even suggesting that designers should delay fast responses deliberately in order to make them more consistent with occasional slow responses. Perhaps such a reduction in response time variability may be desirable. If system response time is **always** slow, a user can adapt to the situation and find something else useful to do while waiting. But surely a better solution is to make all responses uniformly fast, or, where that is not possible, to provide a quick interim message to warn the user that a response will be delayed, as suggested above. In that way, a slow response is made predictable even though it is not consistent with other responses.

Ensuring fast response is probably a greater problem in the design of general-purpose, multi-user, time-shared systems than for dedicated information system applications. Where an information system is designed to accomplish defined tasks in a specified manner, data processing loads can usually be anticipated sufficiently well in user interface design to provide adequately fast response for all transactions.

Consistency is important in all aspects of user interface design. Anyone who has tried to learn a foreign language knows the difficulty of learning irregular verbs, whose conjugation does not follow any consistent rule. In the same way it is difficult to learn (and remember) irregular features of user interface design.

Data entry can be guided by consistent formatting of entry fields on the display, including consistent wording of labels, consistent placement of labels with respect to entry fields, and consistent demarcation of the fields themselves. Some kind of highlighting is frequently recommended for field delineation, displaying field location and boundaries clearly to a user. Even more guidance could be provided by consistent use of different field marking to indicate different types of data entry.

For data display, formats should be consistent from one frame to another, always including a title at the top, labels to indicate page numbers in related displays, standard labeling of control options, standard positioning of guidance messages, etc. Messages indicating user errors should be carefully worded to be both concise and informative. They should also be consistently worded from one message to the next, and consistently located in the display format. Even such a subtle feature as cursor positioning should receive consistent treatment in the software logic of user interface design.

For sequence control, consistent user interface design is even more important. One design feature that can help guide the user is consistent provision of a general

OPTIONS display, a "home base" to which the user can return from any point in a transaction sequence in order to select a different transaction. As a basic principle of user guidance, the interface designer should not rely on a user to remember what prior actions have been taken, or even what action is currently being taken. Users may be distracted by competing job demands. For control actions whose consequences are contingent on context, an indication of context should always be displayed, even when that context was defined initially by the user.

Although consistent interface design will provide much inherent guidance, it is often desirable to include in computer-generated displays some explicit instructions or prompts for the user. Such instructions should be consistently located in display formatting, perhaps always at the bottom where they can be ignored by experienced users. For novice users it may be desirable to provide supplementary guidance or job instruction, available in response to user requests for help. Such optional guidance will adapt the interface to different user capabilities, supporting the novice user without hindering the expert.

The concern here is with on-line user instruction, as opposed to off-line system documentation. The need for on-line instruction has been emphasized by Brown, Brown, Burkleo, Mangelsdorf, Olsen and Perkins in their user interface guidelines developed as an in-house design standard at Lockheed:

> Much of the information commonly provided in paper documentation, such as user manuals, should also be available on line. A manual may not be available when it is needed. Some users may never receive the relevant documentation. They may not know what documents are available, which ones are relevant, or how to procure them. Even users who possess the appropriate documentation will not necessarily have it with them when it is needed.
>
> (1983, page 6-1)

One might add that even if users have the appropriate documentation on hand, they may not be able to find answers to their questions, especially when the documentation is bulky. Effective on-line instruction and other forms of user guidance can reduce the need for off-line training courses based on system documentation.

Certainly there is a strong trend in current system design toward on-line documentation for user instruction. This trend reflects the decreasing cost of computer memory, and also the increasing use of computers by people with little understanding of computer function. The novelty of early pioneering efforts to program computers to teach their own use (Morrill, 1967; Morrill, Goodwin and Smith, 1968; Goodwin, 1974) has now become almost commonplace, as we have learned more about the techniques of on-line aiding.

One of the principal arguments for on-line documentation is economic. Limanowski (1983) estimates a potential cost saving of 70 to 80 percent if on-line documentation can be provided as an alternative to more traditional paper documentation. If that is true, we can expect to see increasing recourse to on-line documentation for that reason alone.

Introduction

People who are responsible for writing instructional material may be the first to note inconsistencies in user interface design. In the documentation of user guidance, whether intended for on-line display or printed handbooks, every special feature and smart shortcut provided by software designers will add an extra paragraph, or sentence, or footnote to the instructional material. As special features proliferate, instructions to the user must expand accordingly. On-line guidance will require more display space, and printed manuals will grow fatter.

From this reasoning, we may conclude that interface design may be improved if its documentation begins early, when changes are still relatively easy to make. On-line documentation can certainly be prepared (and changed) more quickly than printed user manuals, which suggests that early on-line documentation may help interface designers as well as the eventual system users.

Preparation of user guidance material will serve to indicate the point of diminishing returns in further elaboration of user interface software. If it takes ten minutes for a user to learn a special procedure that might save ten seconds in a seldom-used transaction, then design elaboration has outpaced user needs. Considering questions of user guidance throughout interface design will ensure that a sensible balance is struck between efficiency of operation and ease of learning, between functionality and usability. This practical trade-off has been noted by others concerned with user interface design (e.g., Goodwin, 1982), but requires continuing emphasis.

There is clearly an intimate relation between user guidance and interface design. The argument presented above that preparation of user guidance might help improve interface design can also be reversed. That is to say, interface designers can often help to improve user guidance. Preparation of user guidance should not be left solely the separate responsibility of technical writers who were not involved in the design process. Interface designers should participate as well.

In the course of design, a designer must sometimes make decisions that favor one group of users at the expense of another. As an example, for a system that will be used frequently by most of its users, a designer might choose to emphasize flexibility over simplicity when those two design goals conflict, while understanding that this priority could handicap some people who will use the system only occasionally. This priority would be reflected in the designer's choice of dialogue type, in the availability of control options, in display layouts, etc. A thoughtful designer can sometimes predict that users will have difficulty with particular design features. When that is the case, the designer's insight can contribute to the development of effective user guidance material.

In considering guidelines for design of user guidance functions, we must recognize that user guidance is a pervasive concern in interface design. Many of the guidelines proposed for data entry, data display, and sequence control functions have the implicit objective of making a system easier to learn and easier to understand during its use. From general recommendations for design consistency, through more detailed guidelines to help distinguish different aspects of information handling transactions, there must be an underlying concern for guiding the user's interaction with the computer. Thus almost any guideline for user guidance could be cross-referenced to a wide range of other design recommendations. Of the many possible cross references, a number of specific interest are cited in the guidelines proposed here.

Objectives:

Consistency of operational procedures

Efficient use of full system capabilities

Minimal memory load on user

Minimal learning time

Flexibility in supporting different users

Functions

User guidance refers to error messages, alarms, prompts, and labels, as well as to more formal instructional material.

Standard Procedures •1

Design standard procedures for accomplishing similar, logically related transactions.

COMMENT: Standard procedures will facilitate user learning and efficient system operation.

COMMENT: A designer might argue that for one particular transaction a standard procedure does not seem efficient. Perhaps the standard procedure requires one or two more keystrokes than some special procedure that might be devised. But every special feature of interface design will put a small added burden on the user's memory, and where special procedures are not remembered they may not be used properly. Standard procedures will increase **overall** operational efficiency.

REFERENCE: BB 1.2.1, 2.1.5; Reisner, 1981.

SEE ALSO: 3.0•6, 3.0•7, 6.0•7.

Explicit User Actions •2

Require users to take explicit actions to specify computer data processing; the computer should **not** take extra (and possibly unrecognized) actions beyond those specified by a user.

EXCEPTION: Automatic cross file updating following data change might be considered an exception to this rule.

COMMENT: Explicit actions, even though they may require an extra keystroke or two, will help a user to learn procedures and to understand better what is happening in any transaction. In effect, requiring the user to take action to accomplish something can be regarded as a form of guidance.

COMMENT: An interface designer, with expert knowledge of the system and its internal workings, is sometimes tempted to provide the user with "smart shortcuts", where the computer will execute automatically some action that the user would surely need to take. Incorporating such smart shortcuts in interface design, though done with the intention of helping the user, will risk confusing any but the most expert users.

REFERENCE: MS 5.15.2.1.4.

SEE ALSO: 1.0•9, 1.7•4, 3.0•5, 3.1.4•9, 3.2•1, 6.0•9.

•3 **Separate LOG-ON Procedure**

In applications where users must log on to the system, design LOG-ON as a separate procedure that is completed before a user is required to select among any operational options.

COMMENT: Separate LOG-ON will focus user attention on the required input(s), without the distraction of having to anticipate other decisions, and will help reduce initial confusion, particularly for novice users.

REFERENCE: BB 4.6.1.

•4 **Display of Guidance Information**

In general, follow recommendations for the design of data displays when designing user guidance displays.

COMMENT: Some of the specific guidelines for data display are restated for convenient reference in this section, as particularly appropriate for display of user guidance. Many other applicable data display guidelines are cited by cross reference.

SEE ALSO: Section 2.

•5 **Only Necessary Information Displayed**

Tailor the display for any transaction to the current information requirements of the user, so that only relevant data are displayed.

COMMENT: When this can be done successfully, so that only relevant data are displayed, the display itself provides implicit guidance, showing what data should be considered. Conversely, display of irrelevant data will tend to confuse the user.

REFERENCE: BB 1.7; MS 5.15.3.1.2.

SEE ALSO: 2.0•1, 2.0•2, 2.7.2•1.

•6 **Consistent Display Format**

Create display formats with a consistent structure evident to the user, so that any particular type of data is always presented in the same place and in the same way.

COMMENT: Consistent display formats will help new users learn to interact efficiently with the system.

REFERENCE: EG 2.3; MS 5.15.3.1.1.

SEE ALSO: 1.4•24, 1.4•25, 2.0•6, 2.5•1, 2.7.1•4, 3.0•8, 3.1.3•8, 3.1.3•32, 3.1.5•2, 3.4•7, 4.4•8.

▸ Consistent Format for User Guidance •7

Format each different type of user guidance consistently across displays.

EXAMPLE: Display titles might be centered at the top of the display, with display identification codes at the upper left corner. The bottom line of the display should be reserved for command entries, where needed, in which case the line just above it could be used for prompts and advisory messages.

COMMENT: Types of user guidance include display titles, labeling of data entry fields, prompts for data/command entry, error messages, alarms, status and other advisory messages, as well as on-line instructional material.

COMMENT: Consistent allocation of particular areas of a display for user guidance may be sufficient. Certain types of guidance, however, such as alarms and error messages, may require auxiliary coding to help attract user attention.

REFERENCE: BB 1.1.1, 1.1.2, 2.1.2; EG 2.3; PR 4.5.3.

SEE ALSO: 1.4•17, 2.5•11.

Distinctive Format for User Guidance •8

Design display formats so that user guidance material is readily distinguishable from displayed data.

COMMENT: Consistent location of user guidance on the display will usually suffice, but other formatting conventions may help distinguish particular categories of user guidance, such as labels, prompts, etc., as recommended in other guidelines.

REFERENCE: BB 1.8.5, 2.1.1; EG 2.1, 2.3, 3.2.3.

SEE ALSO: 1.4•16, 1.5•2, 2.2•8, 2.5•2, 3.1.3•20, 3.1.4•17.

•9 ▶ **Distinctive Cursor**

Design cursors (in terms of shape, blink, or other means of highlighting) so that they are readily distinguished from other displayed items.

COMMENT: When a cursor is automatically positioned under computer control, it can serve to direct the user's attention to a particular point on a display, and so it should be designed to catch the eye. Even when the cursor has been positioned by a user, if the user is momentarily distracted then a distinctive format may help locate the cursor.

COMMENT: A cursor is the most immediate and continuously available form of user guidance, since it will generally mark the current focus of user attention. With this in mind, the interface designer may decide to use different cursor formats to denote different operational conditions. If that is done, each of those different cursors should be distinctive from other displayed items, and from each other.

SEE ALSO: 1.1•1, 4.1•5.

•10 **Clear Control Labels**

Label function keys and other controls clearly to indicate their function.

REFERENCE: BB 4.4.7; MS 5.15.2.3.9.

SEE ALSO: 1.0•10, 3.1.4•4.

•11 **Clear Data Labels**

Label all displayed data clearly.

COMMENT: Labels for individual data fields can be omitted only where display format and labeling of grouped data clearly identify subordinate items, as in row/column labeling of tabular data.

REFERENCE: BB 1.8.7; MS 5.15.3.1.9.

SEE ALSO: 1.4•5, 1.4•19 thru •21, 1.4•23, 1.5•3, 2.2•3.

•12 **Highlighting Critical User Guidance**

Whatever methods are used to highlight critical items in data display, adopt similar methods to highlight the display of critical user guidance information.

COMMENT: Alarms and warning messages may require output of auxiliary auditory signals as well as display highlighting, to help assure that they attract the user's attention.

REFERENCE: EG 2.1; MS 5.15.3.3.1.

SEE ALSO: 2.6•1, 3.6•2, 4.3•17.

Consistent Coding Conventions •13

Ensure that symbols and other codes have consistent meanings from one display to another.

COMMENT: This practice will aid user learning of new codes, so that they will gain familiarity. Where codes have special meanings, those should be defined in the display.

REFERENCE: BB 3.6.1.

SEE ALSO: 2.6•7, 2.6•16, 2.6•32, 3.1.3•13, 4.4•21.

▶ Familiar Coding Conventions •14

Ensure that codes and abbreviations for data entry/display conform to conventional usage and user expectations.

COMMENT: Conventional usage will aid user learning of codes, and reduce the likelihood of user error in code generation (entry) and code interpretation (display).

COMMENT: Deviation from familiar meanings, such as using an aircraft symbol to denote artillery and vice versa, would almost certainly confuse users.

REFERENCE: BB 3.7.1.

SEE ALSO: 2.6•5, 2.6•32.

Consistent Wording •15

Ensure that the names for function keys, command names, etc., are consistent for similar or identical functions in different transaction sequences.

EXAMPLE: As a negative example, do **not** call the same function EDIT in one place, MODIFY in another, UPDATE in a third.

COMMENT: Consistency in interface design is the fundamental basis of effective user guidance.

REFERENCE: BB 3.7.2; EG 4.2.9.

SEE ALSO: 3.0•10, 3.1.3•12, 3.1.3•14, 3.1.3•19, 3.1.5•7.

4.0 General

•16 ▶ **Familiar Wording**

When wording labels, prompts and user guidance messages, adopt terminology familiar to users.

EXAMPLE:

(Good) `Data requires special access code;`
`call Data Base Admin, X 9999.`

(Bad) `IMS/VS DBMS private data;`
`see DBSA, 0/99-99.`

COMMENT: User testing and iterative design will often be needed to eliminate difficult words, abbreviations and acronyms that are not generally familiar to all users.

REFERENCE: BB 3.7.1, 3.7.3; EG 3.4.5, 4.2.12; PR 2.4; Magers, 1983.

SEE ALSO: 2.0•4, 2.0•12, 3.1.5•6.

•17 ▶ **Task-Oriented Wording**

Adopt task-oriented wording for labels, prompts and user guidance messages, incorporating whatever special terms and technical jargon may be customarily employed in the users' tasks.

COMMENT: Jargon terms may be helpful, if they represent the jargon of the **user** and not of the designer or programmer. The rule here should be to know the users and adapt interface design to their vocabulary instead of forcing them to learn new wording.

REFERENCE: BB 3.7.3; EG 4.2.13; PR 2.4; Magers, 1983.

SEE ALSO: 1.4•22, 2.0•12, 3.1.5•6, 3.1.5•7, 3.2•9.

▶ Wording Consistent with Control Entry •18

Choose wording for user guidance that is consistent with the words used for control entries.

EXAMPLE:

(Good) `To delete a paragraph, press DELETE and then PARAGRAPH.`

(Bad) `To erase a paragraph, press DELETE and then PARAGRAPH.`

EXAMPLE: If a user must complete a control form to specify printer settings, the words used as labels on that form should also be used in any error messages and HELP displays which may guide that process.

COMMENT: When selecting or composing control entries, a user will tend to mimic the vocabulary, format, and word order used in computer displays, including labels, error messages, HELP displays, etc. If displayed wording is consistent with required entries, a user will be more likely to make a correct entry on the first try.

COMMENT: Consistent wording of user guidance will be particularly helpful for dialogues based on constrained natural language. If a designer begins by determining which words and formats users are likely to choose naturally, and then reinforces that usage by incorporating such wording in user guidance, much of a user's interaction with the computer will be predictable. Therefore, the "natural language" need not accommodate the full range of possible entries, but only those entries which users are likely to make.

REFERENCE: Good, Whiteside, Wixon and Jones, 1984; Mooers, 1983; Zoltan-Ford, 1984.

SEE ALSO: 2.0•7, 3.0•13, 3.1.7•1.

▶ Speaking Directly to Users •19

Choose wording for user guidance that speaks directly to a user, rather than talking about users.

EXAMPLE:

(Good)

`Press ENTER to continue.`

(Bad)

`The user should press ENTER to continue.`

REFERENCE: Pakin and Wray, 1982.

•20 ▸ **Affirmative Statements**

Adopt affirmative rather than negative wording for user guidance messages.

EXAMPLE:
(Good)
`Clear the screen before entering data.`
(Bad)
`Do not enter data before clearing the screen.`

COMMENT: Affirmative statements are easier to understand. Tell the user what to **do** rather than what to avoid.

REFERENCE: BB 3.8.3, 5.3.9.

SEE ALSO: 2.1•16.

•21 ▸ **Active Voice**

Adopt active rather than passive voice in user guidance messages.

EXAMPLE:
(Good)
`Clear the screen by pressing RESET.`
(Bad)
`The screen is cleared by pressing RESET.`

COMMENT: Sentences in active voice are easier to understand.

REFERENCE: BB 3.8.5.

SEE ALSO: 2.1•17.

•22 ▸ **Temporal Sequence**

When user guidance describes a sequence of steps, follow that same sequence in the wording of user guidance.

EXAMPLE:
(Good)
`Enter LOG-ON sequence before running programs.`
(Bad)
`Before running programs, enter LOG-ON sequence.`

REFERENCE: BB 3.8.6.

SEE ALSO: 2.1•18.

▶ Consistent Grammatical Structure •23

Be consistent in grammatical construction when wording user guidance.

EXAMPLE:

(Good)	(Bad)
Options:	Options:
s = Select data	s = Select data
e = Erase display	e = Erasure function
w = Write file	w = Write file

COMMENT: Even minor inconsistencies can distract a user, and delay comprehension as the user wonders momentarily whether some apparent difference represents a **real** difference.

COMMENT: Consistent grammatical construction may help a user resolve an ambiguous message (e.g., `Numeric entry`) to understand whether it recommends an action (e.g., "You should enter a number") or indicates an error condition (e.g., "You entered a number when you shouldn't have").

REFERENCE: BB 3.8.4; Pakin and Wray, 1982; Smith, 1981b.

SEE ALSO: 2.0•15, 2.2•5, 3.1.3•11.

Flexible User Guidance •24

When techniques adopted for user guidance (display of option lists, command prompting, etc.) may slow an experienced user, provide alternative paths or modes permitting a user to by-pass standard guidance procedures.

COMMENT: Multiple paths, such as command entry to by-pass a menu, or use of abbreviated rather than complete commands, can speed the performance of an experienced user. The interface designer, however, should take care that such shortcuts supplement rather than supplant the standard, fully guided procedures provided for novice users.

REFERENCE: BB 4.5.

SEE ALSO: 3.0•2, 4.4•31, 4.4•32.

•25 ▸ Easy Ways to Get Guidance

Allow users to switch easily between any information handling transaction and its associated guidance material.

EXAMPLE: Guidance might be displayed as a temporary "window" overlay on the working display, which a user could request or suppress at will.

COMMENT: If user guidance is difficult to obtain, and/or if asking for guidance will disrupt a current transaction (e.g., erase a working display), then users will prefer to guess at proper procedures rather than seeking help.

REFERENCE: Limanowski, 1983.

•26 Speech Output

Consider computer-generated speech output for user guidance messages in environments with low ambient noise, when a user's attention may not be directed toward a visual display or when providing a visual display is impractical.

EXAMPLE: Computer-generated speech might be used to provide prompts or status messages, including warnings. A familiar example of the use of computer-generated speech for warnings is the "talking dashboard", which tells a car driver when a door is open, or when the car requires service.

COMMENT: A noisy environment, particularly noise from other voices, will make it difficult for a user to hear computer-generated speech.

COMMENT: Auditory signals such as computer-generated speech are useful for notifying a user of important information when his/her attention is focused somewhere other than a visual display, such as when a touch-typist transcribes data from a paper form.

COMMENT: Speech output might also help a user who must access a computer from a remote location, by telephone.

COMMENT: When considering speech output for user guidance, remember that people other than the user might hear those spoken messages. Speech output may prove distracting to other people trying to work nearby. Or the user of a system may not wish others to hear his/her messages, as might be the case if spoken messages were provided for an automated banking application.

REFERENCE: Thomas and Rosson, 1984.

▶ Limited Number of Spoken Messages •27

Limit computer-generated speech to provide only a few messages.

EXAMPLE: As negative examples, computer-generated speech would **not** be useful if many messages might be given at one time, or for conveying a lengthy list of menu options.

COMMENT: When messages are spoken, the user must remember each message. If many different messages are given one after another, then a user would probably not remember them all, and might only remember one or two.

▶ Simple Spoken Messages •28

When using computer-generated speech to provide messages, ensure that those messages are short and simple.

COMMENT: If a user does not understand a written message, s/he can reread it. That is not the case with spoken messages. Though a REPEAT function might be provided, a better solution is to restrict use of speech outputs for short and simple messages.

COMMENT: If a user who may not be watching a display must be given long or complex messages, it is probably better to provide a simple auditory signal such as a chime, and then display the messages visually for the user to read. In general, users will understand complex messages better when they see them displayed than when they hear them.

REFERENCE: Thomas and Rosson, 1984.

▶ Distinctive Spoken Warnings •29

If computer-generated speech is used to provide warnings as well as other forms of user guidance, ensure that spoken warnings are easily distinguishable from routine messages.

EXAMPLE: Speech output used to identify dangerous conditions might use some distinctive voice (perhaps female rather than male, or vice versa) and/or preface each warning message with some other distinctive auditory signal.

COMMENT: In some applications, computer-generated speech might be useful for providing a few short and simple warnings. However, if speech output is also used for other purposes, then the warning messages must be distinctive.

REFERENCE: Hakkinen and Williges, 1984; Simpson, McCauley, Roland, Ruth, and Williges, 1985; Simpson and Williams, 1980.

SEE ALSO: 2.6•42.

4.1 Status Information

> *Status information on current data processing should be available at all times, automatically or by request.*

•1 **Indicating Status**

Provide some indication of system status to users at all times.

COMMENT: In some applications, system status may be continuously displayed. Status display can be explicit (e.g., by message), or can be implicit (e.g., by a displayed clock whose regular time change offers assurance that the computer link is still operating). Alternatively, system status information might be provided only on user request, following a general or specific query.

COMMENT: Status information is particularly needed, of course, when system operation is unreliable for any reason. Under those conditions, if status information is not provided by design, users will often devise their own repertoire of harmless but time-wasting inputs to test system performance.

COMMENT: When system status changes, it may be desirable for the computer to generate an advisory message to draw users' attention to that change.

REFERENCE: BB 4.3; MS 5.15.1.4.b.

•2 **Automatic LOG-ON Display**

When users must log on to a system, display appropriate prompts for LOG-ON procedures automatically at a user's terminal; do **not** require users to take any special action to obtain a LOG-ON display, other than turning on the terminal.

COMMENT: An automatic LOG-ON display will signal the operational availability of a terminal, as well as prompting the user to make necessary initial inputs.

REFERENCE: BB 4.6.1; EG 4.2.6.

SEE ALSO: 4.0•3.

▶ LOG-ON Delay •3

If a user tries to log onto a system and LOG-ON is denied because of system unavailability, display an advisory message to tell the user what the system status is and when the system will become available.

EXAMPLE: `System is down for maintenance until 9:30 AM.`

COMMENT: Avoid "as soon as possible" messages. Make an estimate of system availability, and update the estimate later if that becomes necessary.

REFERENCE: BB 4.3.5.

Keyboard Lock •4

If at any time the keyboard is locked, or the terminal is otherwise disabled, notify the user.

EXAMPLE: Control lockout might be signaled by disappearance of the cursor from the display, or by a notable change in the shape of the cursor, accompanied by an auditory signal.

COMMENT: An auditory signal will be especially helpful to touch-typists, who may be looking at source documents for data entry rather than at the display or keyboard.

SEE ALSO: 3.0•20.

Operational Mode •5

When the results of user action are contingent upon different operational modes, then clearly indicate the currently selected mode.

EXAMPLE: A change in display caption and/or cursor shape might suffice to alert users to changing modes.

REFERENCE: BB 4.3.4; MS 5.15.7.5.b.

SEE ALSO: 3.4•4, 4.2•8, 4.4•13.

Other Users •6

When task performance requires data exchange and/or interaction with other users, allow a user to obtain status information concerning other people currently using the system.

COMMENT: If there are **many** other users, it might be helpful to allow a user to ask whether any particular individual is a current user.

SEE ALSO: 5.3•5.

4.1 Status Information

•7

System Load

When task performance is affected by operational load (e.g., number of on-line users), allow a user to obtain status information indicating current system performance, expressed in terms of computer response time.

COMMENT: It may be necessary to define a "standard" function for which computer response time is predicted on a normalized basis.

COMMENT: Such load information is primarily helpful, of course, when system use is optional, i.e., when a user can choose to defer work until low-load periods. But load status information may help in any case by establishing realistic user expectations for system performance.

•8

External Systems

When task performance requires data exchange and/or interaction with other systems, allow a user to obtain relevant status information for external systems.

SEE ALSO: 5.3•5.

•9

Date and Time Signals

When task performance requires or implies the need to assess currency of information, annotate displays with date-time signals.

COMMENT: Depending on the application date-time status might be displayed continuously or periodically on displays that are automatically updated, or by user request.

•10

Alarm Settings

When alarm signals are established on the basis of logic defined by users, permit users to obtain status information concerning current alarm settings, in terms of dimensions (variables) covered and values (categories) established as critical.

COMMENT: Alarm status information will be particularly helpful in monitoring situations where responsibility may be shifted from one user to another ("change of watch").

SEE ALSO: 3.6•1.

> *Routine feedback should be provided by a computer to its users as transactions are processed.*

Consistent Feedback •1

Ensure that every input by a user will consistently produce some perceptible response output from the computer; when a terminal is in use, its display screen should never be blank.

EXCEPTION: A user might choose to blank a display screen temporarily, perhaps to protect data from casual onlookers, but even then some acknowledging message should probably appear, e.g., `Display temporarily suppressed.`

COMMENT: Keyed entries should appear immediately on the display. Function key activation or command entries should be acknowledged either by evident performance of the requested action, or else by an advisory message indicating an action in process or accomplished. Inputs that are not recognized by the computer should be acknowledged by an error message.

COMMENT: Absence of system response is **not** an effective means of signaling acceptable entry. At best, a dialogue without feedback will be disconcerting to the user, as when we talk to an unresponsive human listener. At worst, the user may suspect system failure, with consequent disruption and/or termination of the interaction sequence.

REFERENCE: BB 4.3, 4.3.3, 5.1; EG 3.3.2, 4.2.5, 6.3.7; MS 5.15.2.1.2, 5.15.4.1.12, 5.15.4.1.13; Magers, 1983.

SEE ALSO: 1.0•3, 1.0•12, 1.0•13, 3.0•14, 3.0•16, 3.1.3•9, 3.1.4•10, 6.2•6.

Fast Response •2

Ensure that computer response to user entries will be rapid, with consistent timing as appropriate for different types of transactions.

REFERENCE: MS 5.15.1.8; Shneiderman, 1984; Stewart, 1980.

SEE ALSO: 1.1•5, 2.7.1•6, 3.0•18, 3.1•2.

4.2 Routine Feedback

•3

Feedback for Control Entries

Provide some indication of transaction status whenever the complete response to a user entry will be delayed.

COMMENT: After making an entry to the computer, the user needs feedback to know whether that entry is being processed properly. Delays in computer response longer than a few seconds can be disturbing to the user, especially for a transaction that is usually processed immediately. In such a case some intermediate feedback should be provided, perhaps as an advisory message that processing has been initiated, and ideally with an estimate of how long it will take to complete.

COMMENT: Indicating the progress of computer processing is particularly important when response time is inconsistent and may be lengthy, which is often the case when users perform complex functions on time-shared systems. Displaying time-to-completion or some other indication of progress will permit users to plan their time more effectively, and perhaps to perform other tasks while waiting.

COMMENT: Note that a routine advisory (e.g., `Wait for processing`) displayed before every computer response, whether fast or slow, is **not** an effective indication of transaction status. Users will come to ignore routine messages that are sometimes true but sometimes just false alarms.

REFERENCE: BB 4.3.1; EG 4.2.5; MS 5.15.2.1.3; Myers, 1985.

SEE ALSO: 3.0•14.

•4

▶ Indicating Completion of Processing

When computer processing of a user entry has been delayed, inform the user when processing is completed, and provide appropriate guidance for further user actions.

COMMENT: For long delays, interim feedback on processing status before completion may be reassuring to a user. Such follow-up messages, however, should not interfere with current user activities. It may be desirable to reserve a special display window for prompts and advisory messages.

REFERENCE: BB 4.3.2; MS 5.15.5.3.

SEE ALSO: 3.0•15.

•5

Feedback for Print Requests

When user requests for printed output will be handled by a remote printer, give the user an advisory message confirming that a print request is being processed.

REFERENCE: EG 4.2.14.

SEE ALSO: 1.3•29.

Display Identification •6

Provide a unique identification for each display in a consistent location at the top of the display frame.

EXCEPTION: As a possible exception, interface designers may sometimes provide unlabeled displays as "free-form" screens for text entry and other tasks involving display composition by users.

COMMENT: A displayed title may suffice, although a shorter identification code may be helpful for some purposes. The objective is to help the user recognize a display when it appears, to learn interactive sequences stepping from one display to another, and (in some system applications) to request a particular display directly. Display identification will also help both users and interface designers to refer to individual displays in discussion and documentation.

COMMENT: In applications involving menu selection, it may prove helpful to code each display with the string of option selections (letter codes) used to reach that display. This practice is particularly useful in situations where a user can learn to by-pass the menu selection sequence by entering option string codes as a single command to request a familiar data display.

REFERENCE: BB 1.2.3; MS 5.15.3.1.9; PR 4.5.2.

SEE ALSO: 2.5•10, 2.7.1•2 thru •4.

▶ Identifying Multipage Displays •7

When lists or data tables extend beyond the capacity of a single display frame, inform the user that the display is continued in multiple frames.

EXAMPLE: Incomplete lists might be annotated at the bottom as

    Continued on next page

EXAMPLE: For extended data tables, the display title might be annotated as

    Page __ of __

EXAMPLE: For scrolled data, displays might be annotated with the current and concluding locations as

    Line __ of __

EXCEPTION: In special formats such as spreadsheets the partial nature of a display may be self-evident.

COMMENT: As a complementary recommendation, it may also be desirable to conclude completed lists with the annotation

    End of list

unless the list is so short that it obviously does not fill available display space.

REFERENCE: BB 1.9.7; EG 3.4.1; PR 4.5.5.

SEE ALSO: 2.7.2•5, 2.7.2•6.

•8 **Indicating Operational Mode**

When a user (or computer) action establishes a change in operational mode that will affect subsequent user actions, display some continuing indication of current mode.

EXAMPLE: Selection of a DELETE mode in text editing should produce some kind of warning signal on the display, perhaps by a distinctive change in cursor shape.

COMMENT: This practice is particularly helpful when the mode selected is one seldom used.

COMMENT: Display of mode selection will help prevent unintended data loss when the mode is potentially destructive (e.g., DELETE). For destructive modes, it may help if the mode indication is implemented as some sort of distinctive change in the appearance of the cursor, since the cursor is probably the one display feature most surely seen by a user.

REFERENCE: BB 4.3.4; MS 5.15.7.5.a; Foley and Wallace, 1974.

SEE ALSO: 3.4•4, 3.4•5, 4.1•5, 4.4•13, 6.0•15, 6.0•16.

•9 **▶ Indicating Option Selection**

When previously selected options are still operative, display those options either automatically or on user request.

COMMENT: Displaying a cumulative sequence of option selections may help a novice user learn transaction sequences, and may help any user deal with complex transactions.

REFERENCE: EG 3.4.

SEE ALSO: 4.4•13, 4.4•22.

•10 **▶ Indicating Item Selection**

When a user selects a displayed item in order to perform some operation on it, highlight that item on the display.

COMMENT: This practice will provide a routine natural feedback that item selection has been accomplished, and will provide a continuing reminder to the user of just **what** selection has been made.

COMMENT: Highlighting might be accomplished in different ways. Reverse video is commonly employed for this purpose. For a selection among displayed options, the selected option might be brightened.

REFERENCE: EG 2.1.1, 3.1, 3.1.1; MS 5.15.5.6.

SEE ALSO: 1.1•5, 3.1.3•9, 3.4•6.

Feedback for User Interrupt •11

Following user interrupt of data processing, display an advisory message assuring the user that the system has returned to its previous status.

REFERENCE: BB 4.7.

SEE ALSO: 3.3•6.

4.3 Error Feedback

Error feedback should be provided if an error or other unexpected event prevents routine processing.

•1 **Informative Error Messages**

When the computer detects an entry error, display an error message to the user stating what is wrong and what can be done about it.

EXAMPLE:

(Good) `Code format not recognized; enter two letters, then three digits.`

(Bad) `Invalid input.`

COMMENT: Users should **not** have to search through reference information to translate error messages.

COMMENT: Error messages can be regarded as the most important form of system documentation. Well designed error messages will give help to users automatically, at the point where help is most needed.

REFERENCE: BB 5.2.2, 5.3.2, 5.3.8; EG 3.3.1; MS 5.15.5.7, 5.15.7.5; PR 4.12.1; Dean, 1982; Limanowski, 1983; Magers, 1983; Shneiderman, 1982.

•2 ▶ **Specific Error Messages**

Make the wording of error messages as specific as possible.

EXAMPLE:

(Good) `No record for Loan 6342; check number.`

(Bad) `No record for inquiry.`

COMMENT: Specificity will require computer analysis of data processing transactions in context.

REFERENCE: BB 5.3.7; MS 5.15.7.6, 5.15.7.8; PR 4.12.5.1.

▶ Task-Oriented Error Messages •3

Adopt wording for error messages which is appropriate to a user's task.

EXAMPLE:

(Good) `Contract number not recognized; check the file and enter a current number.`

(Bad) `Entry blocked. Status Flag 4.`

COMMENT: Error messages that can be understood only by experienced programmers (and interface designers) will have no value for ordinary users.

REFERENCE: BB 5.3.5; EG 3.3.7; MS 5.15.7.6; Shneiderman, 1982.

SEE ALSO: 2.0•12, 4.0•17.

Advisory Error Messages •4

If a data entry or (more often) a control entry must be made from a small set of alternatives, an error message that is displayed in response to a wrong entry should indicate the correct alternatives.

SEE ALSO: 3.2•5.

•5 Brief Error Messages

Make error messages brief but informative.

EXAMPLE:

(Good) `Entry must be a number.`

(Bad) `Alphabetic entries are not acceptable because this entry will be processed automatically.`

COMMENT: Often a user will recognize that an error has been made, and the message will serve merely as a confirming reminder. In such instances, short error messages will be scanned and recognized more quickly.

COMMENT: For a user who is truly puzzled, and who needs more information than a short error message can provide, auxiliary HELP can be provided either on-line or by reference to system documentation.

COMMENT: If an on-line HELP explanation is not available, a user may have to refer to system documentation for a coded listing of possible errors. Under those circumstances, some designers display each error message with an identifying code, to facilitate rapid reference to documentation. That practice might help experienced users, who would gradually come to recognize the codes.

REFERENCE: BB 5.3.4; EG 3.1.3, 3.3, 3.3.7; PR 2.2, 4.1.2.2; Shneiderman, 1982.

SEE ALSO: 2.1•13, 2.1•14, 4.4•23.

Neutral Wording for Error Messages •6

Adopt neutral wording for error messages; do **not** imply blame to the user, or personalize the computer, or attempt to make a message humorous.

EXAMPLE:

(Good) `Entry must be a number.`

(Bad) `Illegal entry.`

(Bad) `I need some digits.`

(Bad) `Don't be dumber, use a number.`

COMMENT: Error messages should reflect a consistent view that the computer is a tool, with certain limitations that a user must take into account in order to make the tool work properly. If error messages reflect an attitude that the computer (or its programmer) imposes rules, or establishes "legality", the user may feel resentful. If error messages reflect personalization of the computer, as if it were a friendly colleague, a naive user may be misled to expect human abilities the machine does not actually possess. If error messages are worded humorously, any joke will surely wear thin with repetition, and come to seem an intrusion on a user's concern with efficient task performance.

COMMENT: The same considerations apply for the wording of computer-generated prompts and other instructional material.

REFERENCE: BB 5.5.3; EG 3.3.8, 5.3; MS 5.15.7.6; PR 2.2.

Multilevel Error Messages •7

Following the output of a simple error message, permit users to request a more detailed explanation of the error.

COMMENT: A more complete discussion of each error could be made available on-line, perhaps at several levels of increasing detail, supplemented by reference to off-line system documentation if necessary. Successively deeper levels of explanation might then be provided in response to repeated user requests for HELP.

REFERENCE: BB 1.6, 5.4; EG 3.3.

SEE ALSO: 4.4•28.

4.3 Error Feedback

•8 **Multiple Error Messages**

When multiple errors are detected in a combined user entry, notify the user, even though complete messages for all errors cannot be displayed together.

EXAMPLE: `DATE should be numeric`
`+ 2 other errors`

COMMENT: The computer should place the cursor in the data field referred to by the displayed error message, with other error fields highlighted in some way, e.g., by reverse video. There should also be some means for the user to request sequential display of the other error messages if needed.

REFERENCE: BB 5.2.3; PR 4.12.3.

•9 ▶ **Indicating Repeated Errors**

If a user repeats an entry error, there should be some noticeable change in the displayed error message.

EXAMPLE: A simple expedient might be to display the same verbal message but with changing annotation, perhaps marked with either one asterisk or two.

COMMENT: If an error message is repeated identically, so that displayed feedback seems unchanged, the user may be uncertain whether the computer has processed the revised entry.

•10 **Non-Disruptive Error Messages**

The computer should display an error message only after a user has completed an entry.

EXAMPLE: An error message should **not** be generated as wrong data are keyed, but only after an explicit ENTER action has been taken.

COMMENT: In general, the display of error messages should be timed so as to minimize disruption of the user's thought process and task performance.

REFERENCE: EG 7.1.

SEE ALSO: 1.7•3, 1.7•7.

Appropriate Response Time for Error Messages •11

Display an error message approximately 2-4 seconds after the user entry in which the error is detected.

EXCEPTION: For type-ahead systems with experienced users, error messages should be displayed as quickly as possible.

COMMENT: Longer delays in error feedback may cause user uncertainty or confusion. Longer delays may also cause frustration if the user is already aware of the error, which is often the case.

COMMENT: Shorter delays in error feedback can pose problems of a different sort. An error message following immediately upon a user entry can be disconcerting. Immediate error feedback can also be irritating. User expectations are conditioned by human conversation, where an immediate contradiction is considered rude, and where a polite listener will pause for a few moments before saying that you are wrong.

COMMENT: If error messages take somewhat longer to appear than the routine computer response, then that additional delay may cue the user to expect an error message and pay attention to it.

REFERENCE: EG Table 2.

SEE ALSO: 3.0•18.

Documenting Error Messages •12

As a supplement to on-line guidance, include in the system documentation a listing and explanation of all error messages.

COMMENT: Developing good error messages may require review by both designers and users. Documentation of the complete set of error messages will facilitate such review.

COMMENT: Documentation of error messages will permit users to reference particular messages for fuller explanation, and to review all messages as a means of understanding data processing requirements and limitations.

REFERENCE: BB 5.3.1, 5.4, 5.8.

Cursor Placement Following Error •13

In addition to providing an error message, mark the location of a detected error by positioning the cursor at that point on the display, i.e., at that data field or command word.

COMMENT: Displaying the cursor at a non-routine position will help emphasize that an error has occurred, and direct the user's attention to the faulty entry.

REFERENCE: BB 5.2.5; PR 4.12.1.

SEE ALSO: 4.4•16.

•14 **Displaying Erroneous Entries**

When an entry error has been detected, continue to display the erroneous entry, as well as an error message, until corrections are made.

COMMENT: The error itself may provide useful information, in conjunction with the error message, helping a user understand the specific nature of the error.

•15 **User Editing of Entry Errors**

Following error detection, require the user to re-enter only that portion of a data/command entry which is not correct.

COMMENT: The user should **not** have to rekey an entire command string or data set just to correct one wrong item.

REFERENCE: BB 5.2.1; EG 4.2.3; MS 5.15.7.1.

SEE ALSO: 3.1.5•23, 3.5•3, 6.0•10, 6.3•10.

•16 **Removing Error Messages**

Ensure that a displayed error message is removed after the error has been corrected; do not continue to display a message that is no longer applicable.

COMMENT: The immediate removal of an error message upon error correction will serve as feedback to a user that the corrected entry is indeed correct.

REFERENCE: BB 5.2.6.

•17 **Cautionary Messages**

When a data or command entry seems doubtful, in terms of defined validation logic, display a cautionary message asking the user to confirm that entry.

EXAMPLE: `Blood pH of 6.6 is outside the normal range; confirm or change entry.`

COMMENT: Feedback to the user can be worded to deal with a range of intermediate categories between a seemingly correct entry and an outright error.

REFERENCE: MS 5.15.7.2.

SEE ALSO: 3.5•8, 3.5•11.

User Confirmation of Destructive Entries •18

Require the user to take some explicit action to confirm a potentially destructive data/command entry before the computer will execute it.

COMMENT: A requirement to take an explicit CONFIRM action will direct user attention to questionable entries and help the user avoid the consequences of thoughtless errors.

COMMENT: What constitutes "potentially destructive" requires definition in the context of each system application.

REFERENCE: BB 5.6; EG 4.2.8; Foley and Wallace, 1974.

SEE ALSO: 3.5•7, 6.0•18, 6.0•20, 6.3•19.

Alarm Coding •19

For conditions requiring (or implying the need for) special user attention, code the alarms (or warning messages) distinctively.

EXAMPLE: Alarm messages might be marked with a blinking symbol and/or displayed in red, and be accompanied by an auditory signal; warnings and error messages might be marked with a different special symbol and/or displayed in yellow.

COMMENT: This practice will help ensure appropriate attention, even when a user is busy at routine tasks.

REFERENCE: BB 1.1.2, 7.7.2, 7.7.3; EG 2.1.3; MS 5.15.3.3.2.

SEE ALSO: 2.6•12, 2.6•32, 2.6•35, 2.6•40, 3.6•2, 6.0•17.

4.4 Job Aids

Job aids should provide users with specific task-oriented guidance for every transaction.

•1 **Guidance Information Always Available**

Ensure that specific user guidance information is available for display at any point in a transaction sequence.

COMMENT: Do not require a user to remember information not currently displayed. The user should not have to **remember** what actions are available, or what action to take next. Human memory is unreliable, and without guidance users can be expected to make errors.

REFERENCE: BB 4.3.6; EG 3.4.4; MS 5.15.4.1.7.

SEE ALSO: 2.0•3, 2.7.2•1, 3.1.3•17, 3.1.3•18, 3.2•4, 3.2•5.

•2 **General List of Control Options**

Provide a general list (menu) of control options that is always available to serve as a "home base" or consistent starting point to begin a transaction sequence.

REFERENCE: BB 4.1, 4.4.5.

SEE ALSO: 3.1.5•12, 3.2•2, 3.2•3.

•3 **Logical Menu Structure**

Display menu options in logical groups.

COMMENT: Logical grouping of menu options will aid user learning and selection among displayed alternatives. It may be necessary to test proposed menus to determine just what structural groupings will seem logical to their intended users.

SEE ALSO: 2.1•23, 3.1.3•22, 3.2•3.

•4 **Hierarchic Menus**

When hierarchic menus are used, organize and label them to guide users within the hierarchic structure.

COMMENT: Users will learn menus more quickly if a map of the menu structure is provided as HELP.

REFERENCE: Billingsley, 1982.

SEE ALSO: 3.1.3•24, 3.1.3•25, 3.1.3•30.

Guidance for Sequence Control •5

At every point in a transaction sequence, provide guidance telling the user how to continue.

EXAMPLE:

(Good) Data are current through March 1986.
Press STEP key to continue.

(Bad) Data are current through March 1986.

REFERENCE: BB 4.2; EG 3.1.2; PR 2.2.

SEE ALSO: 3.0•4, 3.1.3•16, 3.2•12.

▶ Transaction-Specific Option Display •6

If there are control options that are specifically appropriate to the current transaction, indicate those options on the display.

EXCEPTION: Treat control options that are generally available at every step in a transaction sequence (PRINT, perhaps) as implicit options that need **not** be included in a display of specific current options.

COMMENT: Usually space can be found on a working display to remind users of several specific control options that are appropriate to the current transaction. A list of all available options, however, may well exceed display capacity. A user may be expected to remember general options, once they have been learned, without their specific inclusion in a display of guidance information. Perhaps the best design approach is to implement general options on appropriately labeled function keys, which will aid user learning and provide a continuing reminder of their availability.

Prompting Entries •7

Provide advisory messages and other prompts to guide users in entering required data and/or control parameters.

COMMENT: Prompting in advance of data/control entry will help reduce errors, particularly for inexperienced users.

COMMENT: If a default value has been defined for null entry, that value should be included in the prompting information.

REFERENCE: EG 4.2.2, 4.2.4; MS 5.15.4.1.4, 5.15.7.5; PR 4.9.2; Foley and Wallace, 1974.

SEE ALSO: 1.0•24, 1.8•4, 3.1.5•11, 3.2•11, 3.5•5.

•8 ▶ **Standard Display Location for Prompting**

Display prompts for data/command entry in a standard location, next to the command entry area at the bottom of the display.

COMMENT: As an alternative, prompts might be provided in a window overlay added to a working display at user request.

SEE ALSO: 4.0•6, and Section 2.7.5.

•9 ▶ **Consistent Format for Prompts**

Use consistent phrasing and punctuation in all prompts.

EXAMPLE:

(Good)

`Save as new file or Overwrite old file (S/O):`

and

`Create new file or Edit old file (C/E):`

(Bad)

`(S)ave as new file or (O)verwrite old file:`

and

`Would you like to create a new file or edit an old file (C/E):`

REFERENCE: Pakin and Wray, 1982.

•10 ▶ **Standard Symbol for Prompting Entry**

Choose a standard symbol for prompts indicating that an entry is required, and reserve that symbol only for that purpose.

EXAMPLE: (Good) `Enter completion code:`
(Bad) `Enter completion code`

COMMENT: Some standard prompting symbol in data entry forms, in menus, in command entry lines, etc., will help to cue users that an input is required. That standard symbol, used along with other formatting cues, will help to alert a user to differences between advisory messages and messages requiring an input.

REFERENCE: BB 2.5.2.

SEE ALSO: 1.4•9, 3.1.3•15.

▶ Concise Wording of Prompts •11

Use concise wording for prompts; eliminate extraneous words.

EXAMPLE:
(Good) `Delete what:`
(Bad) `What text would you like to delete:`

REFERENCE: Pakin and Wray, 1982.

▶ User-Requested Prompts •12

When users vary in experience, which is often the case, provide prompting as an optional guidance feature that can be selected by novice users but can be omitted by experienced users.

COMMENT: Flexibility in prompting can also be provided by multilevel HELP options, so that additional guidance information can be obtained if the simple prompt is not adequate.

SEE ALSO: 3.1.5•11, 3.1.5•13, 4.4•28.

Displayed Context •13

When the results of a user entry depend upon context established by previous entries, display some indication of that context to the user.

EXAMPLE: When the effects of user entries are contingent upon different operational modes, indicate the current mode.

EXAMPLE: If the user is editing a data file, display both the file name and an indication of EDIT mode.

REFERENCE: EG 4.2.1; MS 5.15.4.1.5, 5.15.7.5.

SEE ALSO: 2.7.3•6, 2.7.3•7, 3.0•9, 3.1.4•5, 3.1.4•11, 3.4•1, 3.4•4, 3.4•5, 4.1•5, 4.2•8.

▶ Maintaining Context for Data Entry •14

In a transaction involving extended data entry, display a cumulative record of any previous inputs that are relevant to the current input.

EXAMPLE: In a multipage data entry display, do not rely on the user to remember data accurately from one page to the next.

4.4 Job Aids

•15 **Cues for Prompting Data Entry**

Provide cues for data entry by formatting data fields consistently and distinctively.

EXAMPLE: A eolon might bc used eonsistently to indicate that an entry can be made, followed by an underseored data field to indicate item size, sueh as

Enter part code: ___ ___ ___ - ___ ___

or perhaps just simply

Part code: ___ ___ ___ - ___ ___

COMMENT: Consistent use of prompting eues ean sometimes provide suffieient guidance to eliminate the need for more explicit advisory messages.

REFERENCE: BB 2.1.5; EG 6.3.1.

SEE ALSO: 1.4•10 thru •12, 1.4•18.

•16 **Consistent Cursor Positioning**

As the last step in generating a display output, ensure that the computer will automatically position the cursor so that it appears in a consistent display location for each type of transaction.

EXAMPLE: For data entry displays, the eursor should be plaeed initially at the first data field, or else at the first wrong entry if an crror has been deteeted; in other displays, the cursor might be placed at a eonsistent HOME position, or at the first control option for menu selection, or else in a general eommand entry area, depending upon the type of display.

COMMENT: Consistent eursor positioning will provide an implicit eue for user guidanee.

REFERENCE: EG 4.2.3; MS 5.15.2.1.8.3; PR 3.3.3.

SEE ALSO: 1.1•20, 1.1•21, 1.4•28, 3.2•6, 3.2•7, 4.3•13.

•17 **On-Line System Guidance**

Provide reference material describing system capabilities and procedures available to users for on-line display.

COMMENT: Many systems are not utilized effectively beeause users do not fully understand system capabilities. On-line aeeess to a deseription of system strueture, eomponents and options will aid user understanding.

COMMENT: On-line guidance ean supplement or in some instanees substitute for off-line training. An investment in designing user aids may be repaid by reduced costs of formal training as well as by improved operational performanee.

REFERENCE: BB 6.1, 6.2; Limanowski, 1983; Shneiderman, 1982.

▶ Index of Data •18

In applications where the user can choose what stored data to display, provide an on-line data index to guide user selection.

COMMENT: The data index should indicate file names, objects, properties, and other aspects of file structure that might be used to access different categories of data. It may help to allow users to specify what they require in this index. Some users might wish information on file size, currency (time of latest update), etc. Other users might wish to add a short description of each data file to remind themselves of its contents.

REFERENCE: BB 6.2.

SEE ALSO: 3.1.6•3.

▶ Index of Commands •19

In applications where a user can employ command entry, provide an on-line command index to guide user selection and composition of commands.

COMMENT: Such a command index may help a user to phrase a particular command, and will also be generally helpful as a reference for discovering related commands and learning the overall command language.

REFERENCE: BB 6.2, 6.3; Magers, 1983.

▶ Dictionary of Abbreviations •20

Provide a complete dictionary of abbreviations used for data entry, data display, and command entry, both for on-line user reference and in design documentation.

COMMENT: In applications where users can create their own abbreviations, as in the naming of command "macros", it will be helpful to provide aids for users to create their own individual on-line dictionaries.

REFERENCE: BB 6.5; MS 5.15.6.5.

SEE ALSO: 2.0•21.

Definition of Display Codes •21

When codes are assigned special meaning in a particular display, include a definition of those codes in the display.

COMMENT: This practice will aid user assimilation of information, especially for display codes that are not already familiar.

REFERENCE: BB 7.6.1.

SEE ALSO: 2.6•6.

•22 **Record of Past Transactions**

Allow users to request a displayed record of past transactions in order to review prior actions.

REFERENCE: EG 4.2.7.

SEE ALSO: 3.4•3, 4.5•3.

•23 **HELP**

In addition to explicit aids (labels, prompts, advisory messages) and implicit aids (cueing), permit users to obtain further on-line guidance by requesting HELP.

COMMENT: It is difficult for an interface designer to anticipate the degree of prompting that may be required to guide all users. Moreover, even when prompting needs are known, it may be difficult to fit all needed guidance information on a display page. One possibility would be to provide user guidance in a window overlay added to a working display when a user requests HELP. If more extensive user guidance is needed, then a separate, full-screen HELP display might be provided.

REFERENCE: BB 4.4.3; 6.3; MS 5.15.7.5; PR 3.3.15.

SEE ALSO: Section 2.7.5.

•24 **▶ Standard Action to Request HELP**

Provide a simple, standard action that is always available to request HELP.

EXAMPLE: HELP might be requested by an appropriately labeled function key, or perhaps by keying a question mark into a displayed entry area.

COMMENT: A user should be able to request HELP at any point in a transaction sequence. The procedure should always be the same, whether the user wants an explanation of a particular data entry, a displayed data item, or a command option.

REFERENCE: BB 4.4.3; Keister and Gallaway, 1983.

•25 **▶ Task-Oriented HELP**

Tailor the response to a HELP request to task context and the current transaction.

EXAMPLE: If a data entry error has just been made, HELP should display information concerning entry requirements for that particular data item.

EXAMPLE: If an error in command entry has just been made, HELP should display information concerning that command, its function, its proper structure and wording, required and optional parameters, etc.

REFERENCE: BB 6.3; Magers, 1983.

▶ Clarifying HELP Requests •26

When a request for HELP is ambiguous in context, the computer should initiate a dialogue in which the user can specify what data, message or command requires explanation.

EXAMPLE: The computer might ask a user to point at a displayed item about which HELP is requested.

▶ Synonyms for Standard Terminology •27

When a user requests HELP on a particular topic, the computer should accept synonyms for standard system terminology.

EXAMPLE: If a DELETE command exists, then an explanation of that command might be displayed when a user requests HELP for ERASE.

COMMENT: Users will often attempt to get HELP for a function they know exists, but cannot remember its correct name.

COMMENT: Likely synonyms can be identified by compiling a list of function names used in other similar systems. Other synonyms can be discovered through user testing. It might be desirable to let HELP facilities grow to meet user needs by allowing a system administrator to add synonyms to the system.

▶ Multilevel HELP •28

When an initial HELP display provides only summary information, provide more detailed explanations in response to repeated user requests for HELP.

COMMENT: It is necessarily a matter of judgment just what information should be provided in response to a HELP request. Designing the HELP function to provide different levels of increasing detail permits users to exercise some judgment themselves as to just how much information they want.

REFERENCE: BB 5.4, 6.3.

SEE ALSO: 4.3•7.

▶ Browsing HELP •29

Permit users to browse through on-line HELP displays, just as they would through a printed manual, to gain familiarity with system functions and operating procedures.

REFERENCE: Cohill and Williges, 1985.

4.4 Job Aids

•30 **On-Line Training**

Where appropriate, provide an on-line training capability to introduce new users to system capabilities and to permit simulated "hands on" experience in data handling tasks.

COMMENT: On-line simulation, using the same hardware, user interface software and data processing logic as for the real job, can prove an efficient means of user training. Care must be taken, however, to separate the processing of simulated data from actual system operations.

REFERENCE: BB 6.4; Shneiderman, 1982.

SEE ALSO: 6.0•6, 6.3•21.

•31 ▶ **Flexible Training**

Anticipate the needs of different users, and offer different levels of training for on-line job support.

EXAMPLE: In systems supporting different user jobs, on-line instruction might describe the procedures for each different data handling task.

EXAMPLE: Instruction on keyboard use and lightpen selection of menu options might be provided for novice users, while a tutorial on command language might be provided for more experienced users.

SEE ALSO: 3.0•3, 3.1•1, 3.1.3•35, 3.1.5•4, 4.0•24.

•32 ▶ **Adaptive Training**

In applications where a user must learn complex tasks, design computer-mediated training to adapt automatically to current user abilities.

COMMENT: Adaptive training will require some means for computer assessment of appropriate components of user performance.

SEE ALSO: 4.0•24, 4.5•1.

User records will permit assessment of performance and improvement of user interface design.

User Performance Measurement •1

In applications where skilled user performance is critical to system operation, provide automatic computer recording and assessment of appropriate user abilities.

COMMENT: Recording **individual** performance may be constrained by other considerations, as noted elsewhere in this section.

SEE ALSO: 4.4•32.

Notifying Users •2

Inform users of any records kept of individual performance.

COMMENT: Informing users concerning the nature and purpose of performance records is required by ethical principle, and in some situations may be required by law.

COMMENT: Recording individual performance is potentially subject to abuse, and requires careful scrutiny to ensure essential protection of user privacy. Designers must conform to whatever legal (or otherwise agreed) restrictions may be imposed in this regard.

Transaction Records •3

Ensure that the computer can maintain records of user transactions.

COMMENT: Record keeping might include duration, sequencing and frequency of different transactions. Such transaction records will aid task analysis, particularly in developing systems where data handling requirements are not yet fully defined.

COMMENT: A buffered store of current transactions may be required for user guidance, and for other purposes such as supporting an UNDO capability.

COMMENT: In some applications, transaction recording might be made optional, under control of a system administrator.

SEE ALSO: 4.4•22.

4.5 User Records

•4 **Data Access Records**

Ensure that the computer can maintain records of data access, i.e., which data files, categories, or items have been called out for display.

COMMENT: Records of data use may help software designers improve file structure, reduce data access time, and manage multiple use of shared data files.

COMMENT: Data access records may also be required for purposes of data protection/security.

SEE ALSO: 6.2•8.

•5 **Records of Program Use**

Ensure that the computer can maintain records of use for different portions of application software.

COMMENT: In some cases "program calls" can be derived from transaction records rather than having to be measured directly.

COMMENT: Records of software use may not affect user interface design directly, but can help detect and correct programming inefficiencies and improve system response, particularly during early stages of system development.

•6 **Error Records**

Provide a capability for recording user errors.

COMMENT: Error recording might be done continuously, or by periodic sampling under the control of a system administrator.

COMMENT: Error records can be used to indicate supplemental instruction needed by different users, if individual user errors are identified. In that case, ethical considerations (and in some instances legal considerations) dictate that users be informed that such records will be kept.

COMMENT: Error records can be used to indicate whether particular transactions are giving trouble to many users, in which case design improvements to the user interface may be needed, including changes to user guidance.

COMMENT: For an individual user, the computer might be programmed to generate a special on-line HELP sequence to guide the correction of repeated errors.

REFERENCE: BB 5.7.

SEE ALSO: 4.5•2, 4.6•1.

HELP Records •7

Provide a capability for recording user requests for HELP.

EXCEPTION: There is probably no need to record user browsing of HELP information, if such a capability is provided.

COMMENT: HELP records can be used to detect deficiencies in in early system development, and can be used to improve user guidance in later system operation. In effect, user requests for HELP might be regarded as a possible symptom of poor interface design. If HELP requests are frequent for a particular transaction, then some design improvement may be needed, in procedures, or prompting for user guidance, or both.

SEE ALSO: 4.4•23, 4.4•29.

4.6 Design Change

Design change of software supporting user guidance functions may be needed to meet changing operational requirements.

•1 **Flexible Design for User Guidance**

When user guidance requirements may change, which is often the case, provide some means for users (or a system administrator) to make necessary changes to user guidance functions.

COMMENT: User guidance functions that may need to be changed include those represented in these guidelines, namely, changes in status information, routine and error feedback, job aids, and user records.

COMMENT: Many of the preceding guidelines in this section imply a need for design flexibility. Much of that needed flexibility can be provided in initial interface design. Some guidelines, however, suggest a possible need for subsequent design change, and those guidelines are cited below.

COMMENT: In some applications, it may prove helpful to allow individual users to reword and/or add their own notes to on-line guidance material, just as they might annotate paper documentation.

REFERENCE: Limanowski, 1983.

SEE ALSO: 4.3•12, 4.4•3, 4.4•20, 4.4•27, 4.5•3 thru •7.

•2 **Notifying Users of Design Changes**

When changes are made to interface design, including changes to user guidance functions and on-line documentation, inform users of those changes.

COMMENT: An on-line "message board" appearing at LOG-ON may suffice to notify users of current changes. But more extensive measures may be needed, including corresponding changes to user guidance information, e.g., prompts, error messages, HELP displays, etc.

REFERENCE: Limanowski, 1983.

Section 5

DATA TRANSMISSION

DATA TRANSMISSION

Data transmission refers to computer-mediated communication among system users, and also with other systems. Preceding sections of this report have dealt with the basic functions of using on-line information systems — entering data into a computer, displaying data from a computer, controlling the sequence of input-output transactions, with guidance for users throughout the process. What other functions can a computer serve? One area that increasingly demands our attention is the use of computers for communication, i.e., to mediate the transmission of data from one person to another.

In considering data transmission functions, we must adopt a broad perspective. Data that are transmitted via computer may include words and pictures as well as numbers. And the procedures for data transmission may take somewhat different forms for different system applications.

Data might be transmitted by transferring a data file from one user to another, perhaps with an accompanying message to indicate that such a file transfer has been initiated. Data might be transmitted by directly linking two display terminals, so that whatever data one user keys onto a display will be displayed to another user as well. More commonly, however, data are transmitted as "messages", which involves creation of a specially-formatted file with a formal header to specify the sender, recipient, addresses, etc. In these guidelines, the word "message" is mostly used in this sense, to denote data transmitted with a formal header. But the phrase "message handling" sometimes refers more generally to user participation in data transmission.

In a designed information system, data transmission by file transfer may be largely automatic, accomplished with little user involvement. Data transmission by direct linking would presumably be rare, and achieved only under operational constraints, as when a supervisor links to a subordinate's terminal for reviewing and directing work. Thus most of the guidelines here pertain to data transmission accomplished via formatted messages.

In some applications, computer-mediated data transmission may be a discrete, task-defined activity. Perhaps a system used for planning/scheduling is later used to generate and transmit orders to implement a plan. In such a system, a user can "shift gears", first creating a plan and then transmitting that plan.

In other applications, data transmission may be a continuing, intermittent activity mixed with other tasks. As an example, air traffic controllers might use their computer facilities to exchange information (and to hand over responsibility) while they are performing other essential flight monitoring tasks.

An even broader requirement for computer-based message handling can be seen in systems whose explicit, primary purpose is to support communication (Uhlig, 1981; Smith, 1984). In such applications, computer-mediated data transmission is sometimes called "electronic mail". In conjunction with other new technology, the current development of electronic mail has brought forecasts of a "wired society" in which we become ever more dependent on computers for communication (Martin, 1978).

Effective communication is of critical importance in systems where information handling requires coordination among groups of people. This will be true whether communication is mediated by computer or by other means. Computer-based message handling may offer a potential means of improving communication efficiency, but careful design of the user interface will be needed to realize that potential.

In the interests of efficiency, much data transmission among computers is designed to be automatic, representing a programmed message exchange between one computer and another, with no direct user involvement. If a user does not participate in data transmission, then there is of course no need to include data transmission functions in user interface design. Only when the users themselves are involved in data transmission transactions will interface design guidelines be needed.

For the users of computer systems, data transmission can impose an extra dimension of complexity. A user not only must keep track of transactions with the computer, but also must initiate and monitor data exchange with other people. Users will need extra information to control data transmission, perhaps including status information about other systems, and the communication links with other systems. Users will need feedback when sending or receiving data. Users may need special computer assistance in composing, storing and retrieving messages, as well as in actual data transmission. And users will wish to control the disposition of received messages, perhaps renaming a message and storing it with other related messages, and/or sending it on to other users.

When data transmission functions can be designed as an integral part of an information system, there is a clear opportunity for the interface designer to ensure compatibility of procedures. For example, if the system provides procedures for text editing, file storage, and retrieval in support of so-called "word processing" functions, then those same procedures should serve to edit, store, and retrieve messages. Users will not have to learn a different set of commands, or select from a different assortment of menu options.

In some current applications, however, data transmission functions have been grafted onto an existing system. There is a practical advantage in buying a separate package of message handling software for use in conjunction with an already existing system. The message handling functions do not have to be designed from scratch, and they can often be added without any fundamental redesign of the existing system capabilities.

There is a potentially serious disadvantage, however, in trying to combine separately designed software packages: they will almost surely look different to a user, unless care is taken to design an integrated user interface. Differences in user interface logic can sometimes be "papered over" by the software designer, perhaps by providing a menu overlay that effectively conceals inconsistencies of control logic (Goodwin, 1983; 1984). How well the user interface design has integrated the disparate software packages should then be evaluated in operational use (Tammaro, 1983).

Introduction

Whether the user interface to data transmission functions can be designed from scratch, or is designed separately and then must be integrated with other system functions, some guidelines may be needed to help the interface designer. Recent studies of computer-based message handling have been chiefly concerned with determining the functional capabilities required in data transmission (cf., Goodwin, 1980). There is already evidence, however, that the practical use of data transmission functions can be limited by deficiencies in user interface design (Goodwin, 1982).

The general objectives of user interface design in other functional areas are equally valid for data transmission functions. Procedures for data transmission should be consistent in themselves, and consistent with procedures for data entry and display. Interface design should minimize effort and memory load on the user, and permit flexibility in user control of data transmission.

Objectives:

Consistency of data transmission

Minimal user actions

Minimal memory load on user

Compatibility with other information handling

Flexibility for user control of data transmission

Functions

Data transmission refers to computer-mediated communication among system users, and also with other systems.

Functional Integration •1

Ensure that data transmission functions are integrated with other information handling functions within a system.

COMMENT: A user should be able to transmit data using the same computer system (and procedures) used for general entry, editing, display, and other processing of data.

COMMENT: A user should **not** have to log off from a general data processing system and log on to some other special system in order to send or receive a message. If data transmission facilities are in fact implemented as a separate system, that separation should be concealed in user interface design, so that a user can move from general information handling to message handling without interruption.

Functional Wording •2

Choose functional wording for the terms used in data transmission — for preparing and addressing messages, for initiating and controlling message transmission and other forms of data transfer, and for receiving messages — so that those terms will match users' work-oriented terminology.

EXAMPLE: A user should be able to address messages to other people or agencies by name, without concern for computer addresses, communication network structure and routing.

COMMENT: In general, a user should **not** have to learn the technical details of communication protocols, codes for computer "handshaking", data format conversion, etc., but should be able to rely on the computer to handle those aspects of data transmission automatically.

REFERENCE: Bruder, Moy, Mueller and Danielson, 1981.

SEE ALSO: 3.1.5•3, 3.1.5•5.

•3 **Consistent Procedures**

Ensure that procedures for preparing, sending and receiving messages, are consistent from one transaction to another, and are consistent with procedures for other information handling tasks.

EXCEPTION: Data transmission that does not involve formal messages might by-pass standard procedures as in the direct linking of terminals, or might require special procedures as in the transfer of data files.

COMMENT: Procedures should be the same for handling different kinds of messages and for messages sent to different destinations, although procedures for handling high-priority messages might incorporate special actions to ensure special attention.

COMMENT: Users should be able to use the same procedures to enter, edit and display messages as they use to enter, edit and display other kinds of data. Computer-generated error messages and other forms of user guidance should also be consistent from one kind of information handling to another.

SEE ALSO: 4.0•1.

•4 **Minimal Memory Load on User**

Design the data transmission procedures to minimize memory load on the user.

EXAMPLE: Interface software might provide automatic insertion into messages of standard header information, distribution lists, etc.

EXAMPLE: The computer should provide automatic queuing of outgoing messages pending confirmation of transmission, and of incoming messages pending their review and disposition.

EXAMPLE: Software might provide automatic record keeping, message logging, status displays, etc.

•5 **Minimal User Actions**

Design the data transmission procedures to minimize required user actions.

EXAMPLE: In some applications, software logic might prepare and transmit messages automatically, derived from data already stored in the computer.

EXAMPLE: Software logic might provide automatic reformatting of stored data for transmission, where format change is required.

EXAMPLE: Interface software might provide automatic insertion into messages of standard header information, distribution lists, etc.

Control by Explicit User Action •6

Design the data transmission procedures so that both sending and receiving messages are accomplished by explicit user action.

COMMENT: Automatic message generation and receipt will be helpful in many applications, but in such cases the user should be able to monitor transmissions, and should be able to participate by establishing, reviewing and/or changing the computer logic that controls automatic data transmission.

SEE ALSO: 1.0•9, 3.0•5, 4.0•2, 6.0•9.

Flexible User Control •7

Provide for flexible control of data transmission, so that users can decide what data should be transmitted, when, and where.

EXCEPTION: In monitoring and operation control applications, data transmission must often be event-driven.

COMMENT: Flexible control of message handling will help ensure that routine data transmissions will not interfere with a user's other activities.

REFERENCE: Williamson and Rohlfs, 1981.

▶ Interrupt •8

Allow users to interrupt message preparation, review, or disposition, and then resume any of those tasks from the point of interruption.

SEE ALSO: Section 3.3.

Flexible Message Filing •9

In applications requiring general-purpose message handling, provide users with flexible capabilities for filing copies of draft messages during preparation, transmitted messages, and received messages, and organizing those message files.

COMMENT: For most information handling systems, it is probably desirable to design the user interface so that users do not have to concern themselves with the detailed structure of data files. For message handling, however, users will often need to decide themselves whether and where to store transmitted data, i.e., how messages should be organized in their filing. Appropriate computer aids should be provided for message storage and retrieval, to permit naming of message files, grouping of files into larger "folders", and indexing the resulting file structure.

5.0 General

•10 Message Highlighting

Provide software capabilities to annotate transmitted data with appropriate highlighting to emphasize alarm/alert conditions, priority indicators, or other significant second-order information that could affect message handling.

COMMENT: Second-order information, i.e., data about data, will often aid the handling and interpretation of messages. Such annotation might be provided automatically by software logic (e.g., a computer-generated date-time-stamp to indicate currency), or might be added by the sender of a message to emphasize some significant feature (e.g., attention arrows), or by the receiver of a message as an aid in filing and retrieval.

REFERENCE: Williamson and Rohlfs, 1981.

Preparing messages for transmission involves specification of contents, format, and header information.

Message Composition Compatible with Data Entry •1

Ensure that procedures for composing messages are compatible with general data entry procedures, especially those for text editing.

EXCEPTION: In systems where special editing capabilities are available for special tasks, as in some programming systems, users should be able to choose whether a special computer editor will be used for message preparation.

COMMENT: A user should **not** have to learn procedures for entering message data that are different from more general data entry, particularly if those procedures might involve conflicting habits.

SEE ALSO: 5.0•3, and Section 1.

User-Designed Message Formats •2

When required message formats will vary unpredictably, allow users to compose and transmit messages with a format of their own design.

COMMENT: Establishing new formats, particularly if automatic data validation must be defined for specified fields, may require special skills. Therefore this capability might be provided to a system administrator and not to all system users.

▶ Unformatted Text •3

Allow users to compose and transmit messages as unformatted text.

COMMENT: Allowing users to create arbitrary text messages (sometimes called "chatter") will let users deal flexibly with a variety of communication needs not anticipated by system designers.

•4 **Stored Message Forms**

When message formats should conform to a defined standard or are predictable in other ways, provide prestored forms to aid users in message preparation.

EXAMPLE: A stored form might be used to create a routine report for transmission to a standard distribution list.

COMMENT: It may also be desirable to allow users to modify stored forms for their own purposes, and to define and store their own message forms.

SEE ALSO: 5.0•5.

•5 **Automatic Message Formatting**

When data must be transmitted in a particular format, as in data forms or formatted text, provide computer aids to generate the necessary format automatically.

COMMENT: When transmitting data, a user should **not** have to convert those data from whatever format was used originally for data entry.

COMMENT: It is **not** sufficient merely to provide computer checking of formats generated by the user. Computers should help users to avoid errors, and not just to identify errors.

REFERENCE: Deutsch, 1981.

SEE ALSO: 5.0•5.

•6 **▶ Automatic Text Formatting**

When transmitted text must be formatted in a particular way, format control should be automatic with no extra attention required from the user.

EXAMPLE: Header/paging formats might be inserted automatically in preparing text for transmission.

EXAMPLE: Defined message formats might be filled automatically from stored text.

SEE ALSO: 1.3•21, 5.0•5.

▶ Data Forms •7

In preparing data forms for transmission, allow users to enter, review, and change data on an organized display with field labels, rather than requiring users to deal with an unlabeled string of items.

COMMENT: User composition and review of unlabeled data strings, especially those requiring delimiters to mark items, will be prone to error. If such data strings are needed for economy of transmission, they should be generated by the computer automatically from data entered in a form-filling dialogue.

COMMENT: Transmission of data from one computer to another will often be more economical if field labels and other display formatting features are omitted. In such cases, a format code should be included with the message, so that forms filled by the sender can be re-created in a display useful to the receiver.

SEE ALSO: 5.0•3, 5.5•12.

▶ Tables and Graphics •8

In preparing tabular or graphic data for transmission, allow users to enter, review, and change data in customary formats, regardless of what the computer-imposed format will be for actual transmission purposes.

EXAMPLE: Tabular data in messages should be prepared (and later received) in row-column format, even though the table entries might actually be transmitted as a coded string of data items.

SEE ALSO: 5.0•3, 5.5•12.

Flexible Data Specification •9

Provide users with flexible means for specifying the data to be transmitted.

COMMENT: When preparing a message, a user may wish to specify data to be included in the message by selecting a particular file, either all or a designated part, or by defining a data category.

SEE ALSO: 5.0•7.

5.1 Preparing Messages

•10 **▶ Incorporate Existing Files**

Allow users to incorporate an existing data file in a message, or to combine several files into a single message for transmission.

COMMENT: It should **not** be necessary for a user to re-enter for transmission any data already entered for other purposes. It should be possible to combine stored data with new data when preparing messages for transmission.

REFERENCE: Williamson and Rohlfs, 1981.

SEE ALSO: 1.0•1, 5.0•5.

•11 **▶ Incorporate Other Messages**

Allow users to incorporate other messages in a message being prepared for transmission.

EXAMPLE: A user might wish to forward with comments a message received from someone else.

•12 **Variable Message Length**

Allow users to prepare messages of any length.

COMMENT: In particular, data transmission facilities should **not** limit the length of a message to a single display screen or to some fixed number of lines. There will usually be some implicit limit on message length imposed by storage capacity or the amount of time it would take to transmit a very long message. However, a user might sometimes choose to increase storage or accept transmission delays in order to send a long message required by a particular task.

REFERENCE: Bruder, Moy, Mueller and Danielson, 1981.

•13 **Saving Draft Messages**

Allow users to save draft messages during their preparation, or upon their completion.

COMMENT: A user should **not** be forced to recreate a message if its preparation is interrupted for some reason. Users should be able to specify how to save draft messages (i.e., in what file), just as they may decide how to save copies of transmitted and received messages.

SEE ALSO: 5.0•9.

Addressing messages may require user action and computer aids to specify the destinations for data transmission.

Destination Selection •1

Allow users to specify the destination(s) to which data will be transmitted.

EXCEPTION: In some bus communication systems, it might be desirable to permit content-driven communication, where potential recipients can request all messages on particular topics, whether or not those messages are specifically addressed to them.

COMMENT: Specification of message destination might be in terms of system users, as individuals or groups, or other work stations and terminals (including remote printers), or users of other systems. Standard destinations may be specified as a matter of routine procedure, with special destinations designated as needed for particular transactions.

COMMENT: For most applications, it is important that users be able to send a message to multiple destinations with a single transmission action. For multiple recipients, it will usually be helpful to show all addresses to all recipients, so that they will know who else has received the message. In some cases, however, it may be desirable to permit transmission of "blind" copies.

SEE ALSO: 5.0•7.

Standard Address Header •2

For addressing and identifying messages, provide a basic set of header fields that can be interpreted by all systems to which users will send messages.

EXAMPLE: Basic header fields might include DATE, TO, FROM, COPIES, and perhaps some message identification number.

COMMENT: In any particular system, it should be possible for users (or a system administrator) to specify additions to the standard header fields in order to convey more descriptive information about different types of messages. Possible additions to the basic address fields might include SUBJECT, KEYWORDS, and REFERENCES.

REFERENCE: Deutsch, 1981.

•3 **Prompting Address Entry**

When a user must specify the address for a message, provide prompting to guide the user in that process.

COMMENT: Prompting might consist of a series of questions to be answered, or an address form to be completed by the user, or reminders of command entries that may be needed.

SEE ALSO: 4.4•7, 5.0•4.

•4 **Address Directory**

Provide users with a directory showing all acceptable forms of message addressing for each destination in the system, and for links to external systems.

COMMENT: In addition to the names of people, users may need to find addresses for organizational groups, functional positions, other computers, data files, work stations, and devices. The directory should include specification of system distribution lists as well as individual addresses.

REFERENCE: Garcia-Luna and Kuo, 1981; Williamson and Rohlfs, 1981.

•5 ▶ **Aids for Directory Search**

Provide computer aids so that a user can search an address directory by specifying a complete or partial name.

COMMENT: Users will often remember a partial address, even if they cannot remember its complete form.

•6 ▶ **Extracting Directory Addresses**

Allow users to extract selected addresses from a directory for direct insertion into a header in order to specify the destination(s) for a message.

COMMENT: Direct insertion of addresses from a directory will avoid errors which a user might make in manual transcription and entry, as well as being faster.

COMMENT: Users may also wish to extract addresses from the directory in order to build their own distribution lists, or add to a "nickname" file.

User-Assigned Nicknames for Addressing •7

Allow users to define nicknames for formal addresses, to save those nicknames in their own files, and to specify those nicknames when addressing messages.

COMMENT: There are several implications to such a nickname capability. First, a user might wish to assign nicknames to computers and other devices (e.g., printers) as well as to people. Second, if a user defines a nickname, the computer must check to ensure that the nickname is unique in that user's nickname file. Third, nicknames must take precedence over system names when a user addresses a message; i.e., the computer must check the user's nickname file before checking the system-wide address list. Fourth, nicknames should **not** be transmitted; i.e., the computer should automatically transform nicknames into standard system addresses when completing the address header for message transmission.

System Distribution Lists •8

Provide formal distribution lists recognized by the system so that users can specify multiple addresses with a single distribution list name.

EXAMPLE: A formal distribution list might be maintained of people who are working on a particular project, or who are members of a particular organizational group.

COMMENT: Recognized system distribution lists need **not** be expanded to the names of individual addressees when a message is transmitted.

COMMENT: The authority to use system distribution lists may be limited in some cases. For example, not everyone might be permitted to send messages to a distribution list of all employees in a large organization.

REFERENCE: Bruder, Moy, Mueller and Danielson, 1981; Deutsch, 1984; Garcia-Luna and Kuo, 1981; Williamson and Rohlfs, 1981.

▶ Access to Distribution List Information •9

Provide users with information about distribution lists on which they are included, and about those distribution lists which they are authorized to use.

COMMENT: Users should be able to discover the names of all people on distribution lists they are authorized to use.

REFERENCE: Deutsch, 1984.

•10 **Informal Distribution Lists**

Allow individuals or groups to create their own informal distribution lists for local use.

COMMENT: Such informal group and individual distribution lists should be expanded by the computer to show individual addressees prior to message transmission.

COMMENT: As a procedural matter, informal distribution lists shared by a group might be created and maintained by some designated custodian, who could control access to such lists. Whereas any individual user's personal distribution lists might be changed freely.

REFERENCE: Bruder, Moy, Mueller and Danielson, 1981; Garcia-Luna and Kuo, 1981.

•11 **▸ Lists Within Lists**

Within a distribution list, allow users to include other distribution lists as well as individual names.

COMMENT: In providing this capability, note that care must be taken to ensure that computer expansion of nested lists will not cause continuous looping, as in a case where list A includes list B which in turn includes list A.

REFERENCE: Deutsch, 1984.

•12 **▸ Modifying Distribution Lists**

Provide computer aids to permit users to modify distribution lists once created.

COMMENT: Users might sometimes wish to modify their stored distribution lists, or the distribution for any particular message. Appropriate review/change procedures should be provided.

SEE ALSO: 5.0•4, 5.0•5.

•13 **Automatic Expansion of Partial Addresses**

Allow users to enter a partial name when specifying addresses, if that will identify a particular destination uniquely.

COMMENT: If a partial name is ambiguous (i.e., it partially identifies two or more addressees), then a list of the possible addressees might be provided and the user asked to select the correct one.

COMMENT: The computer should automatically expand any partial address into its full form when completing the header for message transmission.

Automatic Address Checking •14

Provide computer checks for address accuracy (i.e., recognized content and format) and require users to correct mistakes before initiating message transmission.

COMMENT: A transmitted message should always have correct address information. In particular, a message sent to several people should have the correct address for all of those people on all copies. If the sender has specified a wrong address then **no** copies of the message should be sent until that error has been corrected. Otherwise, a recipient might attempt to reply using the same inaccurate addresses that were received, and address errors could propagate within the system.

COMMENT: Ideally, address checking should occur as the user enters addresses, when correction may be relatively easy. If that is not possible, and an address error is not found until the user has moved on to some other task, then the user should not be interrupted to correct the error. Instead, a nonintrusive error message should be displayed, so that a correction can be made at the user's convenience.

COMMENT: Sometimes an address error may not be discovered until a user has requested message transmission and logged off the system. In such a case, message transmission should be delayed until that user returns to the system, receives a delayed notification of the address error, and corrects it. That message delay is an acceptable consequence under these circumstances, i.e., when a user leaves the system right after making an error.

SEE ALSO: 1.7•1.

Addressing Replies to Messages Received •15

If a user wishes to reply to a received message, provide the appropriate address(es) automatically, along with a reference to that received message.

COMMENT: The computer should address the reply to the sender of the original message, include some identification of the original message (e.g., its date, time, subject, and/or other identification), and should address copies of the reply to other recipients of the original message.

Editing Address Headers •16

Allow users to edit the address fields in the header of a message being prepared for transmission.

COMMENT: An editing capability will allow users to correct errors, and to change unwanted addresses which may have been supplied automatically by the computer.

COMMENT: Procedures for editing addresses should be the same as those for editing message content.

SEE ALSO: 5.0•3, 5.1•1.

5.2 Addressing Messages

•17 Single Occurrence of Address

Ensure that the address of any particular recipient will occur only once in a message.

COMMENT: Duplicate addresses may confuse users. Within the limits of reasonable cost, computer checking should be provided to remove any address duplication that might result from the overlap of several expanded distribution lists, or from the cumulative listing of recipients of messages replying to replies.

COMMENT: If duplicate addressing cannot reasonably be prevented, it is important to ensure that duplicate copies of a message are not actually sent to any recipient.

COMMENT: There might be a special situation in which a user wants to send multiple copies of a message to a single recipient. In such a case, the extra copies should be specified explicitly in message transmission, rather than achieved indirectly by duplicating that recipient's address.

REFERENCE: Deutsch, 1984.

SEE ALSO: 5.4•4.

•18 Serial Distribution

In applications requiring coordinated review of messages by several recipients, allow the sender to specify a serial distribution so that a message will be passed from one recipient to the next.

COMMENT: Depending on the circumstances, serial recipients might or might not be allowed to annotate a forwarded message with comments.

SEE ALSO: 5.1•11.

•19 Redistributing Received Messages

Allow users to redistribute received messages by enlarging their address headers.

COMMENT: Here redistribution is considered to be forwarding a message without editing or added comment, with only its address being changed. Where this capability is provided, some means must be adopted to indicate that a message has been forwarded verbatim, and to identify who added what names to the original distribution.

SEE ALSO: 5.1•11.

Initiating data transmission should usually be under user control, with computer aids for that process.

User-Initiated Transmission •1

Allow users to initiate data transmission directly, by entering an explicit SEND command.

COMMENT: In some routine reporting situations it might help to initiate message transmission automatically. More often, however, users will wish to initiate transmission themselves. User control of initiation could permit a user to review and edit prepared messages before sending them, and possibly catch some mistakes before they are propagated.

COMMENT: In applications where message review is critical (perhaps for purposes of security), users might be required to take some extra CONFIRM action to release data for transmission.

SEE ALSO: 5.0•6, 5.0•7.

User Review Before Transmission •2

When computer aids are provided for preparing, addressing, and initiating message transmission, allow users to review and change messages prior to transmission.

SEE ALSO: 6.4•2.

▶ Optional Message Display •3

For sending messages, allow users to choose whether to transmit a displayed version or to transmit directly from computer-stored data files.

COMMENT: Message transmission from displays will permit a user to review and edit messages before sending them. But users might sometimes wish to transmit a prepared message directly, without having first to display that message for review.

SEE ALSO: 5.0•7.

•4 **Computer-Initiated Transmission**

When standard messages must be transmitted, as when a computer is monitoring external events and reporting data change, provide computer aids to initiate transmission automatically.

EXAMPLE: Many operations-monitoring tasks might benefit from automatic generation of messages to report routine events.

COMMENT: Automatic transmission of routine messages will reducc user workload and help ensure timely reporting. However, users should be able to monitor automatic message initiation, and may sometimes wish to modify initiation logic. Appropriate review/change procedures should be provided, perhaps under control of a system administrator.

•5 **Information About Communication Status**

Allow users access to status information concerning the identity of other system users currently on-line, and the availability of communication with external systems.

COMMENT: Such information may influence a user's choice of destinations and choice of communication methods, as well as the decision when to initiate transmission. For example, a user might choose to link directly with another user who is currently on-line, but might compose a message for deferrcd transmission to an inactive user.

REFERENCE: Dzida, 1981.

SEE ALSO: 4.1•6, 4.1•8.

Sender Identification •6

When a message is sent, the computer should append to it fields showing the sender's address, and the date and time of message creation and/or transmission.

EXCEPTION: Like other formatting recommendations, this guideline would not apply when data transmission may be accomplished without formal messages, i.e., transmission by direct linking (where no formal headers are prepared) or by file transfer (where header information might be sent as a separate message rather than incorporated in the transmitted data file).

COMMENT: Recipients will generally need to know the origin of any received message. For some messages, a simple identification of the sender may be sufficient. In special situations, however, it may be necessary to provide special procedures for authenticating a sender's identity.

COMMENT: For some applications, recipients must know when a message was created, in order to assess the currency of its contents. For other applications, users may need to know when a message was transmitted; its time of creation might not be important.

REFERENCE: Price, 1981.

SEE ALSO: 6.4•6.

Assignment of Priority •7

When messages will have different degrees of urgency, i.e., different implications for action by their recipients, allow the sender of a message to designate its relative priority, or else have the computer assign priority automatically.

COMMENT: The computer might impose limits on the priority that any particular user can assign to messages. In a military system, for example, only certain users might be authorized to send messages at the highest priority levels.

SEE ALSO: 5.5•6.

Automatic Queuing for Transmission •8

Provide automatic queuing of outgoing messages, in order to reduce the need for user involvement in the routine processing of data transmission.

EXAMPLE: The computer might queue outgoing messages when communication channels to some addressees are temporarily unavailable, and then initiate transmission automatically when a link can be established.

COMMENT: Specific requirements will vary with the application, but **some** queuing should be provided.

SEE ALSO: 5.0•4.

•9 **Deferring Message Transmission**

Allow users to defer the transmission of prepared messages, to be released by later action.

EXAMPLE: Prepared messages might be left in some "outbox" file for subsequent transmission when a user has finished an entire batch.

COMMENT: A user might wish to defer data transmission until a batch of related messages has been prepared, or perhaps until some specified date-time for release.

SEE ALSO: 5.0•7.

•10 **▶ Transmission at Specified Date/Time**

Allow users to specify a date and time for transmitting prepared messages.

COMMENT: A user should be able to cancel such a deferred transmission prior to its specified initiation time.

•11 **Return Receipt**

Provide for message transmission with "return receipt requested" if users will require that capability.

COMMENT: The logic of what constitutes message "receipt" might vary from one application to another. Where sender and receiver share a common system, receipt might be defined as occurring when the recipient actually requests display of an incoming message. Where sender and receiver work with different (and perhaps dissimilar) message systems, receipt might be defined more tenuously. For example, a message might be considered "received" when the recipient is merely notified of its arrival, or opens an "in-box" permitting potential access to that message.

COMMENT: Any "registered mail" of this kind should be labeled as such for all recipients of this mail.

SEE ALSO: 5.4•3.

•12 **Cancel Transmission**

Allow senders to cancel a request for transmission of a message that has not yet been sent.

SEE ALSO: 3.3•3, 5.0•7, 5.4•8.

Printing Messages •13

Allow users to print copies of transmitted messages in order to make hard-copy records.

EXCEPTION: In some applications, security constraints might make printed records inadvisable.

COMMENT: This may be regarded as a special case of message addressing, where a message is sent to a printer as well as to other people.

COMMENT: User requirements for printed data are often unpredictable to system designers, and so a general printing capability should be provided.

REFERENCE: EG 4.2.14; MS 5.15.9.2; Bruder, Moy, Mueller and Danielson, 1981.

SEE ALSO: 2.7.1•11, 6.2•7, 6.4•7.

5.4 Controlling Transmission

Controlling transmission can often be handled automatically, but users may need information about that process.

•1 **Automatic Protection of Transmitted Data**

Ensure that transmitted data are protected automatically with parity checks to detect and correct any errors that may occur, and buffering until acknowledgment of receipt.

COMMENT: The computer should check transmitted data to determine whether an error has occurred; correct such errors automatically, if necessary by requesting retransmission; and call to the user's attention any case in which a detected error cannot be automatically corrected.

SEE ALSO: 6.4•1.

•2 **Automatic Feedback**

Provide automatic feedback for data transmission confirming that messages have been sent or indicating transmission failures, as necessary to permit effective user participation in message handling.

COMMENT: Specific information requirements will vary with the application, but some feedback should be provided.

COMMENT: Users might require notification only of exceptional circumstances, as in the event of transmission failure after repeated attempts.

COMMENT: For the electronic equivalent of registered mail, it might be helpful to provide the sender confirmation that a message has been successfully transmitted, or possibly even to notify the sender when a recipient has actually read (displayed) the message.

•3 **▶ User Specification of Feedback**

Allow users to specify what feedback should be provided for routine message transmission, and also to request specific feedback for particular messages.

COMMENT: Users may wish to specify minimal feedback (or perhaps none at all) for routine message transmissions. On the other hand, users may wish to request more specific feedback for transmission of critical messages, as an electronic equivalent of registered mail.

SEE ALSO: 5.3•11.

Send Single Copy •4

Ensure that only one copy of any message will be transmitted to any individual addressee.

COMMENT: Receiving multiple copies of the same message can be confusing to a user, and will impose an unnecessary burden on the review and disposition of received messages.

COMMENT: The problem of duplicative message transmission might arise if the same address should appear in both the TO and COPY fields of a message header, or appear once explicitly and again in some added distribution list(s). Thus one way to avoid message duplication is to ensure only a single appearance of each address in a message. If that is not practicable, then checking logic should be provided to transmit a single message to duplicated addresses.

COMMENT: If for any reason a user **did** want to send multiple copies of a message to a single recipient, that should be specified explicitly in message transmission, rather than being accomplished by duplicate addressing.

SEE ALSO: 5.2•17.

Queuing Failed Transmissions •5

In the event of transmission failure, provide automatic queuing to preserve outgoing messages.

COMMENT: Automatic queuing and retransmission of outgoing messages will reduce the need for user attention. If transmission fails in repeated attempts, however, then user intervention may be required, and some notification of that problem should be given to the user. If transmission fails because of faulty addressing, the user should be notified immediately.

REFERENCE: Dzida, 1981.

SEE ALSO: 5.2•14.

▶ Saving Undelivered Messages •6

If message transmission is not successful, provide automatic storage of undelivered messages.

COMMENT: Transmission failure should not cause loss or destruction of messages, and should not disrupt the sender's work in any other way.

•7 ▶ **Notification of Transmission Failure**

If message transmission is not successful, notify the sender and if possible include an explanation of the problem.

COMMENT: It may help a user to know whether transmission has failed because of faulty addressing, or communication link failure, or some other reason, in order to take appropriate corrective action.

•8 **Message Recall**

Allow users to recall any message whose transmission has been initiated, if it has not yet been received by its addressee(s).

COMMENT: The intent here is to allow users to change their minds, in situations where a message may have been sent by mistake. The difficulty of message recall will vary with the circumstances. If a message is still in the "out-box" (i.e., it has not yet actually been sent), then its recall can be accomplished simply by canceling the transmission request. If a message has been transmitted but not yet actually received (i.e., it still resides unread in a recipient's "in-box"), then perhaps it can still be retrieved. If a message is recalled before its intended recipient has seen it, that recipient need not be notified. If the recipient **has** seen it, then the sender should be notified that the message cannot be recalled. In that case, the message might be canceled or countermanded procedurally, by sending some follow-up message. If a message to several addressees has been seen by some recipients but not by others, then it would be subject only to partial recall; countermanding might be a better solution.

REFERENCE: Hannemyr and Innocent, 1985.

SEE ALSO: 5.3•12.

•9 **Automatic Record Keeping**

When a log of data transmissions is required, maintain that log automatically, based on user specification of message types and record formats.

COMMENT: The objective here is to minimize routine "housekeeping" chores for the user. Once a user has specified the appropriate logging format (i.e., what elements of each message should be recorded), computer aids should generate the log automatically, either for all messages, or for specified categories of messages, or for particular messages identified by the user.

COMMENT: The same kind of aids should be available for maintaining a journal of data transmissions, in applications where a full copy of each message is required.

SEE ALSO: 5.0•4, 5.0•5.

Receiving messages may require computer aids for queuing, reviewing, filing, or otherwise disposing of incoming data.

Specifying Sources •1

For receiving messages, allow users to specify from what sources data are needed, and/or will be accepted.

COMMENT: Source specification might be in terms of data files, or other users, or external sources. Standard sources might be specified as a matter of routine procedure, with special sources designated as needed for particular transactions.

COMMENT: Computer-mediated message handling offers the potential for screening out the electronic equivalent of "junk mail" or "crank calls". A user might choose to be selective in specifying the people or organizations from which messages will be received, or in specifying data categories of interest.

SEE ALSO: 5.0•7.

Specifying Device Destination •2

For receiving messages, allow users to choose the method of receipt, i.e., what device (files, display, printer) will be the local destination.

COMMENT: Device destination might be specified differently for different types of messages, or for messages received from different sources. Transmitted data might be received directly into computer files. Incoming messages might be routed to an electronic display for quick review, and/or to a printer for hard-copy reference.

COMMENT: If a specified receiving device is not operable, such as a printer that is not turned on, the computer should call that to the user's attention.

COMMENT: When messages are received via display, provide queuing of incoming messages so that they will not interfere with use of that display for other information handling tasks.

SEE ALSO: 5.0•7.

5.5 Receiving Messages

•3 **Queuing Messages Received**

Provide default logic so that, unless otherwise specified by a user, the computer will route incoming messages automatically to a queue ("in-box"), pending subsequent review and disposition by the user.

COMMENT: Some computer buffering of received messages will be required for a user who is not logged on, and also to deal with near simultaneous receipt of multiple transmissions. The recommendation here is that the buffer queue for incoming transmissions be enlarged as necessary to permit indefinite retention of messages. Any queue will have limits, of course, and the user should be warned before those limits are exceeded.

REFERENCE: Williamson and Rohlfs, 1981.

•4 **Message Notification at LOG-ON**

When users log on to a system, notify them of any data transmissions received since their last use of the system.

COMMENT: Depending on the application, a user might wish to know that some message has been received, or how many messages, or what kinds of messages.

COMMENT: If automatic notification of received messages is not feasible, allow users to check for message arrival by requesting information from their general data processing system, without having to access some special message handling system for that purpose. At the least, a user should be able to find out how many new messages have arrived.

REFERENCE: Bruder, Moy, Mueller and Danielson, 1981.

•5 **Nondisruptive Notification of Arriving Messages**

For messages arriving while a user is logged on to a system, ensure that notification of message arrival will not interfere with that user's other information handling tasks.

COMMENT: An incoming message should **not** preempt a user's working display. Instead, the computer might indicate message arrival by an advisory notice in a portion of the display reserved for that purpose.

COMMENT: Review and disposition of received messages, like other transactions, should generally require explicit actions by a user. When an incoming message implies an urgent need for user attention, notify the user.

REFERENCE: EG 7.1.

SEE ALSO: 5.0•6, 6.4•5.

Indicating Priority of Received Messages •6

In applications where incoming messages will have different degrees of urgency, i.e., different implications for action, notify recipients of message priority and/or other pertinent information.

COMMENT: If incoming messages are queued so that their arrival will not interrupt current user tasks, which is a good idea, then users should be advised when an interruption is in fact necessary.

COMMENT: Notification of urgent messages might be routed to a special area of a user's working display for immediate reference, whereas notification of routine messages might be deferred, or perhaps routed to a printer for more leisurely review at the user's convenience.

SEE ALSO: 5.3•7.

Filters for Message Notification •7

Allow users to specify "filters" based on message source, type, or content, that will control what notification is provided for incoming messages.

EXAMPLE: A user might wish the arrival of all messages from a particular source to produce a special notification of some sort.

SEE ALSO: 5.0•7.

Warning of Incompatible Format •8

If a message (or other data transmission) arrives in a format incompatible with system decoding and/or device capabilities, advise the intended recipient to take some appropriate action that will not destroy the message itself or any other data.

EXAMPLE: If the user of an alphanumeric terminal should receive a message that includes nondisplayable graphic symbols, then the computer might notify the user and offer partial display of the readable portions of the message.

COMMENT: In some instances a recipient might be able to resolve a format problem by changing the device destination specified for a particular message.

COMMENT: Failure of message delivery should not disrupt any other work of the intended recipient. For example, the arrival of some message with an unrecognized format, or the attempted delivery of a graphic message at a text-only terminal, should not cause any system failure. Such an undeliverable message might be saved in a system file for subsequent disposition. Or that message (or some notification) might be returned to its sender.

SEE ALSO: 5.4•6, 6.0•4, 6.0•17.

•9 **Information about Queued Messages**

Allow users to review summary information about the type, source, and priority of queued incoming messages.

COMMENT: In some applications, a user might need notification only of urgent messages, and rely on periodic review to deal with routine messages. Summary information about queued incoming messages should help guide message review.

REFERENCE: Williamson and Rohlfs, 1981.

•10 **▶ Indicate Message Size**

Include in summary listings and at the beginning of each message some indication of message size.

EXAMPLE: Message size might be calculated as number of lines, and indicated in its header.

•11 **▶ Specifying Format for Message Listings**

Provide some means for users to specify the order in which header fields are displayed in messages and in message summary listings.

EXAMPLE: For different purposes, a user might wish to display messages with the topic on the first line, or with the sender's name on the first line.

EXAMPLE: A user might wish to scan a summary list of messages grouped by topic, or by sender, or by priority, etc.

COMMENT: Users should be able to assign names to various stored header format specifications of this kind, so that a particular desired header format might be invoked by name.

User Review of Messages in Queue •12

Provide convenient means for user review of received messages in their incoming queue, without necessarily requiring any further disposition action, i.e., without removal from the queue.

EXCEPTION: In some operational applications, user review of critical messages might be accompanied by a requirement for further actions to ensure timely response.

COMMENT: Rapid review of queued messages will permit a user to exercise judgment as to which require immediate attention, and/or which can be dealt with quickly. Other messages may be left in the queue for more leisurely disposition later.

COMMENT: A user with limited facilities for data storage might not be able to accept for immediate filing all of the messages received during a prolonged absence from the system. In such circumstances it seems clear that the user should be able to review received messages **before** deciding which to store in personal files.

Filters for Ordering Message Review •13

Allow users to specify "filters" based on message source, type, or content, that can control the order in which received messages will be read.

COMMENT: The aim here is to facilitate flexible user control over the review of incoming messages. In particular, a user should **not** be constrained to read incoming messages in the order in which they are received.

SEE ALSO: 5.0•7.

Message Review Compatible with Data Display •14

Ensure that computer aids and procedures for reviewing messages are consistent with other system capabilities for general data display.

COMMENT: Users should **not** have to learn procedures for reviewing messages that are different from general data display, particularly if those procedures might involve conflicting habits. Users should be able to page or scroll through a summary list of messages, or a particular displayed message, just as they might manipulate any other data display. Users should be able to select items from a displayed summary list of messages, just as they might select items from a displayed menu of control options.

SEE ALSO: 5.0•3, and Section 2.

5.5 Receiving Messages

•15 Labeling Received Messages

Allow users to assign their own names and other descriptors to received messages.

COMMENT: A user might wish to file received messages for future reference. User-assigned labels could help identify a stored message and distinguish it from other filed messages.

COMMENT: In the absence of labeling by a recipient, the computer might assign by default whatever descriptors have been provided by the message sender.

•16 Annotating Received Messages

Allow users to append notes to a received message, and ensure that such annotation will be displayed so that it will be distinct from the message itself.

COMMENT: In most applications, users should **not** be allowed to make **changes** in received messages. Any such changes would simply provide too much chance for resulting confusion. But users should be able to append, file, and display in some distinctively separate form, their own comments about received messages. If changes are desired in a message itself, then its recipient might make a copy of that message (with appropriate change of its header information) and then edit that copy.

•17 Filters for Message Filing

Allow users to specify "filters" based on message source, type, or content, that will control how messages should be filed.

EXAMPLE: A user might want to file in a single "folder" all messages about a particular topic.

SEE ALSO: 5.0•9.

•18 Discarding Messages

Allow users to discard unwanted messages without filing them, or even without reading them in applications where "junk mail" may be received.

COMMENT: Discarding messages, like other user actions, should be reversible. That is to say, a discarded message should be filed temporarily in some "wastebasket" from which it could later be retrieved if the user has not yet left the system.

REFERENCE: Williamson and Rohlfs, 1981.

> *Design change of software supporting data transmission may be needed to meet changing operational requirements.*

Flexible Design for Data Transmission •1

When data transmission requirements may change, which is often the case, provide some means for users (or a system administrator) to make necessary changes to transmission functions.

COMMENT: Data transmission functions that may need to be changed include those represented in these guidelines, namely, changes in message preparation and addressing, the initiation and control of message transmission, and the handling of received messages.

COMMENT: Many of the preceding guidelines in this section imply a need for design flexibility. Much of that needed flexibility can be provided in initial interface design. Some guidelines, however, suggest a possible need for subsequent design change, and those guidelines are cited below.

SEE ALSO: 5.0•7, 5.1•2, 5.2•1, 5.3•4, 5.4•3, 5.5•1.

Section 6

DATA PROTECTION

DATA PROTECTION

Data protection attempts to ensure the security of computer-processed data from unauthorized access, from destructive user actions, and from computer failure. With increasing use of computer-based information systems, there has been increasing concern for the protection of computer-processed data. Data protection is closely allied with other functional areas. The design of data entry, data display, sequence control, user guidance, and data transmission functions can potentially affect the security of the data being processed. In many applications, however, questions of data protection require explicit consideration in their own right.

Data protection must deal with two general problems. First, data must be protected from unauthorized access and tampering. This is the problem of data security. Second, data must be protected from errors by authorized system users, in effect to protect users from their own mistakes. This is the problem of error prevention.

Design techniques to achieve data security and to prevent user errors are necessarily different, but for both purposes the designer must resolve a fundamental dilemma. How can the user interface be designed to make correct, legitimate transactions easy to accomplish, while making mistaken or unauthorized transactions difficult? In each system application, a balance must be struck between these fundamentally conflicting design objectives.

Concern for data security will take different forms in different system applications. Individual users may be concerned with personal privacy, and wish to limit access to private data files. Corporate organizations may seek to protect data related to proprietary interests. Military agencies may be responsible for safeguarding data critical to national security.

The mechanisms for achieving security will vary accordingly. Special passwords might be required to access private files. Special log-on procedures might be required to assure positive identification of authorized users, with records kept of file access and data changes. Special confirmation codes might be required to validate critical commands.

At the extreme, measures instituted to protect data security may be so stringent that they handicap normal system operations. Imagine a system in which security measures are designed so that every command must be accompanied by a continuously changing validation code which a user has to remember. Imagine further that when the user makes a code error, which can easily happen under stress, the command sequence is interrupted to re-initiate a user identification procedure. In such a system, there seems little doubt that security measures will reduce operational effectiveness.

In recent years, computer security measures have concentrated increasingly on **automatic** means for data protection, implemented by physical protection of computing equipment and by tamper-proof software. Automation of security makes good sense. If data security can be assured by such means, there will be less need to rely on fallible human procedures. And, of course, user interface design will be that much easier.

It seems probable, however, that absolute data security can never be attained in any operational information system. There will always be **some** reliance on human judgment, as for example in the review and release of data transmissions, which will leave systems in some degree vulnerable to human error. Thus a continuing concern in user interface design must be to reduce the likelihood of errors, and to mitigate the consequences of those errors that do occur.

Like data security, error prevention is a relative matter. An interface designer cannot reasonably expect to prevent **all** errors, but frequent user errors may indicate a need for design improvement. Data protection functions must be designed 1) to minimize the entry of wrong data into a system; 2) to minimize mistakes that make wrong changes to stored data; and 3) to minimize the loss of stored data. In considering these objectives, prevention of catastrophic data loss is vital for effective system operation, but all three aspects of data protection are important.

Data entry and change transactions are, of course, frequently performed by system users. Careful interface design can help prevent many errors in those transactions, by providing automatic data validation and reversible control actions, as described in previous sections of these design guidelines. But the designer needs a good deal of ingenuity in applying guidelines within the context of each data handling job.

The use of job context for computer validation of user inputs is best illustrated by example. As one such example, a local newspaper published the following discussion of data entry error prevention, suggesting ways to reduce billing errors:

> . . . designers of applications systems have resorted to a number of strategies to minimize ill effects and keep the errors from escaping into the world at large. The commonest of these is the process known as 'verification,' which in its simplest form, means instructing the system to respond to input with the figurative question, 'Do you really mean that?'
>
> That is, when a data[-entry] operator enters an amount, or name, or serial number — the system draws attention to the just-typed item (by causing it to flash on-screen, for example). The operator then is supposed to take a good hard look at the item and press a verification key if the data is correct.
>
> Better yet, what you need is for the system to do some checking on its own . . . to a certain extent, it can use internal evidence (and the percentages) to perform its own verification.
>
> Here's a simple strategy that, though currently used to some extent, will some day become universal, one hopes. It's good because it relies on an understanding of human habits.
>
> Let's take billing again. Most times, when you pay a bill, you pay either the minimum amount due or the full balance. Suppose we instruct the machine to compare the operators entry of 'amount paid' with the minimum due and with the full balance for that particular account. If the entry is equal to one or the other,

> pass it on through. It's very likely correct. If there's a discrepancy, discontinue entry and signal the operator to check the amount.
>
> And it does so optimally: the right ones pass through with minimum slow down, the potential wrong ones get the attention.
>
> (Bertoni, 1982)

Similar reasoning might be applied in other data entry jobs. Once data are correctly entered into a computer, the emphasis shifts to prevention of unwanted changes to the data, including the extreme form of change represented by data loss. Stored data must be protected from unreliable computer operation and also from system users. Advances in computer technology, with less volatile memory, automatic archiving to back-up data stores, and redundant processing facilities, have significantly reduced the hazard of data loss resulting from machine failure. What remains is to reduce the hazard of human failure.

In the interface design guidelines presented here, the primary concern is for the general users of computer systems. But data protection from human error requires consideration in other aspects of system design and operation. In particular, the expert operators who maintain and run the computer system must assume a large responsibility for data protection.

Consider the following example. In one computer center, an operator must enter a command "$U" to update an archive tape by writing a new file at the end of the current record, while the command "$O" will overwrite the new file at the beginning of the tape so that all previous records are lost. A difference of one keystroke could obliterate the records of years of previous work. Has that ever happened? Yes, it has. If an error **can** happen, then it probably **will** happen.

In that respect, expert computer operators are just like the rest of us. When tired, hurried, or distracted, they can make mistakes. And not all computer operators are experts. Some are still learning their jobs, and so may be even more error-prone. Careful design and supervision of operating procedures is needed to minimize data processing errors.

If data loss from machine failure and data loss from faulty system operation are minimized through careful design, then the most serious threat to data protection is the system user. This is especially true in applications where the user must participate directly in establishing and maintaining stored data files. Means must be found to protect files from inadvertent erasure.

Some difficult design trade-offs may be required. As an example, consider a possible design guideline that might say that a user should not be allowed to change or delete data without first displaying the data. In a file deletion transaction it would usually be impractical to force a user to review the entire file. One might imagine displaying just the first page of a file nominated for deletion, and requiring the user to CONFIRM the DELETE action. But even that would be disruptive in many circumstances.

As a fall-back position, we might recommend that when a file has been nominated for deletion enough descriptive information should be displayed about that file so that a reasonably attentive user can determine whether that file **should** be deleted. The issue is how to ensure that the user intends to do what s/he is actually doing.

When a user selects a file for deletion, at least as much information should be provided as when a user selects a file for display and editing. Thus, if an on-line index of displayable files contains a line of information about each onc, perhaps including name, description, size, and currency (date last changed), then such information would also be appropriate in prompting the CONFIRM action for file deletion:

CONFIRM DELETION of the following file:

USIplan, 5-year plan for USI effort, 3 pages, 11-25-83

Any required confirmation procedure, of course, will tend to slow file deletion, in accord with the general guideline that destructive actions should be made difficult. Where is the trade-off when destructive actions are also frequent? What about the user who wishes to scan a file index and delete a series of files? Must each separate DELETE action be confirmed? Unless DELETE actions are easily reversible, the answer for most users is that an explicit confirmation probably should be required for each file deletion. When a user must undertake a series of file deletions, the repetitive nature of the task may increase the risk of inadvertent deletion, and so increase the need for CONFIRM protection.

Explicit DELETE commands are not the only actions that can result in accidental file erasure. In some systems, it is possible to overwrite a stored file with whatever data are currently on-line, in "working" storagc. Used properly, this capability permits desired editing and replacement of files. A user might call out a file, make changes to it, and then store it again under its own name.

Used improperly, this capability risks file dclction. A user might call out and edit a file, but then absent-mindedly store it with the name of another file, thus overwriting whatever data had been previously stored in that other file. Such a hazard requires just as much protection as an outright DELETE action, or perhaps even more since the danger is more subtle. In effect, an explicit CONFIRM action should bc required whenever a user attempts to store a data file under the name of any other file already stored in the system. The prompt for confirmation might read something like this:

CONFIRM OVERWRITING the current file of this name:

SCG, sequence control guidelines, 45 pages, 10-08-83

It is interesting that many systems do not require this kind of selcctivc confirmation. One well-known system requires user confirmation of **every** overwrite action, even in the common case where an edited file is being stored by the same name to replace its previous version. Thus the CONFIRM action itsclf becomes routine, and no longer provides any significant protection to a forgetful user. Another system avoids the problem by the rigid expedient of allowing a user to store an edited file only under its last previous name, which is safe but sometimes inconvenient.

To some extent a wary user can protect her/himself by careful selection of file names, trying to ensure that any file name is descriptive of file content, and also distinctive in comparison with the names of other files. In practice, that goal is hard to achieve. Users often work with groups of files dealing with diffcrent aspects of a common topic. For such files, if names are descriptive they will tend to be similar rather than distinctive. If file names are made longer in order to become more distinctive, then those longer names may reduce efficiency when using file names for storage and retrieval.

In systems where there is no effective on-line protection against inadvertent file deletion or replacement, either a user must be exceedingly careful, or else the system must provide effective off-line procedures to recover from archived records an earlier version of an accidentally erased file. Neither alternative is entirely satisfactory. Even a careful user will make mistakes. And archives will not protect a user from loss of current work.

A better solution can be provided by on-line computer aids to make user actions reversible. In effect, user interface software should be designed to permit users who notice unintended deletions to retrieve lost files by taking an UNDO action. Some current systems provide such an UNDO capability, permitting users to change their minds and to correct their more serious mistakes.

There must be, of course, some practical time limit to the reversibility of data processing. A user might be able to UNDO the last previous deletion, or perhaps even all deletions made during the current working session, but there seems no feasible way to make it easy to undo a particular deletion made days ago and now regretted. Moreover, reversibility will not help a user who does not notice that a mistake has been made. So even where an UNDO capability is available, other aspects of the user interface must still be carefully designed.

The guidelines proposed in the following pages illustrate the range of topics to be considered in this area, and the general need for many kinds of data protection. These guidelines draw heavily from recommendations already made in previous sections, as indicated by extensive cross referencing. In this new context of data protection, some previous guidelines have been slightly reworded, others preserved intact. They are repeated here for the convenience of a designer who must review all material pertinent to data protection functions.

These guidelines do not resolve the fundamental design dilemma discussed above, namely, how to make a system easy to use but hard to misuse. The guidelines do, however, indicate where design decisions must be made.

Objectives:

Effective data security

Minimal entry of wrong data

Minimal loss of needed data

Minimal interference with information handling tasks

Functions

Data protection concerns security from unauthorized use, and potential loss from equipment failure and user errors.

Automated Security Measures •1

Whenever possible provide automated measures to protect data security, relying on computer capabilities rather than on more fallible human procedures.

COMMENT: For protection against unauthorized users, who may be intruders in a system, the need for automated security measures is clear. For legitimate users, the need for data protection is to minimize data loss resulting from potentially destructive equipment failures and user errors. Even careful, conscientious users will sometimes make mistakes, and user interface logic should be designed to help mitigate the consequences of those mistakes.

Warning of Threats to Security •2

Provide computer logic that will generate messages and/or alarm signals in order to warn users (and system administrators) of potential threats to data security, i.e., of attempted intrusion by unauthorized users.

COMMENT: For protecting data from unauthorized use, it may not be enough merely to resist intrusion. It may also be helpful if the computer can detect and report any intrusion attempts. In the face of persistent intrusion attempts, it may be desirable to institute countermeasures of some sort, such as changing uscr passwords or establishing other more stringent user authentication procedures.

REFERENCE: EG 2.1.3; CSC-STD-002-85.

SEE ALSO: 6.1•6.

•3 **Protection from Computer Failure**

Provide automatic measures to minimize data loss from computer failure.

EXAMPLE: Depending upon the criticality of the application, different protective measures may be justified, including periodic automatic archiving of data files, maintenance of transaction logs for reconstruction of recent data changes, or even provision of parallel "backup" computing facilities.

COMMENT: An automatic capability is needed because users cannot be relied upon to remember to take necessary protective measures.

COMMENT: Though not strictly a feature of user interface design, reliable data handling by the computer will do much to maintain user confidence in the system. Conversely, data loss resulting from computer failure will weaken user confidence, and reduce user acceptance where system use is optional.

REFERENCE: MS 5.15.4.6.3.

•4 **Protection from Interference by Other Users**

Protect data from inadvertent loss caused by the actions of other users.

COMMENT: In systems where information handling requires the coordinated action of multiple users, it may be appropriate that one user can change data that will be used by others. But when multiple users will act independently, then care should be taken to ensure that they will not interfere with one another. Extensive system testing under conditions of multiple use may be needed to determine that unwanted interactions do not occur.

COMMENT: When one user's actions can be interrupted by another user, as in defined emergency situations, that interruption should be temporary and nondestructive. The interrupted user should subsequently be able to resume operation at the point of interruption without data loss.

REFERENCE: MS 5.15.4.6.5.

SEE ALSO: 3.0•22.

Protection from Interrupts •5

When a proposed user action will interrupt a current transaction sequence, provide automatic means to prevent data loss; if potential data loss cannot be prevented, warn the user and do **not** interrupt without user confirmation.

EXAMPLE: If a user should interrupt a series of changes to a data file, then the computer might automatically save both the original and the changed versions of that file for subsequent user review and disposition.

COMMENT: Some interrupt actions such as BACKUP, CANCEL, or REVIEW, will by their definition cause only limited data change, and so need no special protection. However, if an interrupt action may cause extensive data change (e.g., RESTART, LOG-OFF), then require the user to confirm that action before processing.

REFERENCE: BB 4.7.

SEE ALSO: 3.3•6.

Segregating Real from Simulated Data •6

When simulated data and system functions are provided (perhaps for user training), ensure that **real** data are protected and real system use is clearly distinguished from simulated operations.

REFERENCE: BB 6.4.

SEE ALSO: 4.4•30, 6.3•21.

Consistent Procedures •7

Provide clear and consistent procedures for different types of transactions, particularly those involving data entry, change and deletion, and error correction.

COMMENT: Consistent procedures will help users develop consistent habits of operation, reduce the likelihood of user confusion and error, and are especially important for any transaction that risks data loss.

REFERENCE: BB 1.2.1, 2.1.5.

SEE ALSO: 4.0•1.

Appropriate Ease or Difficulty of User Actions •8

Ensure that the ease of user actions will match desired ends; make frequent or urgent actions easy to take, but make potentially destructive actions sufficiently difficult that they will require extra user attention.

6.0 General

•9 Control by Explicit User Action

Ensure that the computer changes data only as a result of explicit actions by a user, and does not initiate changes automatically.

EXCEPTION: In an operations monitoring situation, a computer might accept data changes automatically from external sources (sensors), if appropriate software is incorporated to ensure validation and protection of the input data.

EXCEPTION: A computer might perform cross-file updating automatically, following data change by a user.

COMMENT: The aim here is to preserve clarity of system operation for the user. In effect, a computer should not initiate data changes unless requested (and possibly confirmed) by a user. Well-intentioned interface designers are sometimes tempted to contrive "smart shortcuts" in which one user action might automatically produce several other associated data changes, perhaps saving the user a few keystrokes in special cases. If such shortcuts cannot be made **standard** procedures, they will tend to confuse users and thus pose a potential threat to data protection.

SEE ALSO: 1.0•9, 1.1•4, 3.0•5, 3.1.3•6, 3.5•6, 4.0•2, 5.0•6.

•10 User Review and Editing of Entries

For all inputs, whether data entries or commands, allow users to edit composed material before requesting computer processing.

COMMENT: Input editing will allow users to correct many errors before computer processing. When an error is detected, a user will be able to fix it by editing, i.e., without having to retype any correct items (which might introduce further errors).

REFERENCE: BB 5.2.1; EG 5.4; Neal and Emmons, 1984.

SEE ALSO: 1.4•2, 3.5•2, 4.3•15.

•11 Disabling Unneeded Controls

When function keys and other devices are not needed for current control entry, and especially when they may have destructive effects, disable them temporarily under software control so that they cannot be activated by a user.

COMMENT: Some means should also be provided to help users distinguish currently active from disabled controls, such as brightening (active) or dimming (disabled) their associated labels. If labeling is adequate, then user selection of a disabled control need produce no response. If adequate labeling cannot be provided, then user selection of a disabled control should produce an advisory message that the control is not currently active.

SEE ALSO: 3.1.4•12, 3.2•10.

▸ Protecting Physical Controls •12

If activation of function keys (and other control devices) may result in data loss, locate them separately and/or physically protect them to reduce the likelihood of accidental activation.

REFERENCE: MS 5.15.4.1.2.

SEE ALSO: 3.1.4•18.

Safe Defaults •13

If automatic defaults are provided for control entries, ensure that those defaults will protect against data loss, or at least **not** contribute to the risk of data loss.

EXAMPLE: When requesting a printout of filed data, one control option might be to delete that file after printing; the default value for such a destructive option should automatically be set to NO whenever the printing options are presented to a user for selection.

Safe Response to Random Inputs •14

Ensure that user interface software will deal appropriately with all possible user errors and random inputs, without introducing unwanted data change.

COMMENT: The interface designer must try to anticipate every possible user action, including random keying and perhaps even malicious experimentation. The user interface should be "bullet-proofed" so that an unrecognized entry at any point will produce only an error message and will **not** change stored data.

REFERENCE: BB 5.1; MS 5.15.2.1.2; PR 4.12.4.5.

SEE ALSO: 3.5•1.

Explicit Action to Select Destructive Modes •15

Require users to take explicit action to select any operational mode that might result in data loss; the computer should **not** establish destructive modes automatically.

EXAMPLE: In text editing, if a user takes a DELETE action, that in itself should not establish a continuing DELETE mode.

COMMENT: In many applications, it may be better not to provide **any** destructive mode. Instead of providing a DELETE mode, for example, require that DELETE be a discrete action subject to confirmation by the user when the requested data deletion is extensive. User interface design must determine the proper balance here between data protection and operational efficiency.

SEE ALSO: 1.3•32, 4.2•8.

•16 **Feedback for Mode Selection**

When the result of user actions will be contingent upon prior selection among differently defined operational modes, provide a continuous indication of the current mode, particularly when user inputs in that mode might result in data loss.

EXAMPLE: If a DELETE mode is being used to edit displayed data, some indication of that mode should be continuously displayed to the user.

COMMENT: A user cannot be relied upon to remember prior actions. Thus any action whose results are contingent upon previous actions can represent a potential threat to data protection.

REFERENCE: BB 4.3.4; MS 5.15.5.5.

SEE ALSO: 4.2•8.

•17 **Warning Users of Potential Data Loss**

For conditions which may require special user attention to protect against data loss, provide an explicit alarm and/or warning message to prompt appropriate user action.

SEE ALSO: 3.5•8, 4.3•14, 4.3•19.

•18 **User Confirmation of Destructive Actions**

Require users to take an explicit extra action to CONFIRM a potentially destructive control entry before it is accepted by the computer for execution.

REFERENCE: BB 5.6; MS 5.15.7.5.b; Foley and Wallace, 1974.

SEE ALSO: 1.3•32, 1.3•34, 3.1.5•25, 3.5•7, 4.3•18.

•19 **▶ Distinctive CONFIRM Action**

Provide a distinctively labeled CONFIRM function key for user confirmation of potentially destructive actions.

SEE ALSO: 3.5•9.

▸ Separate CONFIRM Action •20

Require users to wait for computer prompting in order to CONFIRM a potentially destructive action, so that the confirmation will constitute a second, separate action.

EXAMPLE:

```
Enter next command: D____

If deleted, all data in this file will be lost.
Enter YES to confirm deletion: ____
```

COMMENT: No single user action should cause significant change or loss of stored data, such as deleting an entire data file. Requiring users to strike two keys, such as DELETE followed immediately by CONFIRM, is not sufficient protection; such double keying may become habitual. The DELETE and CONFIRM actions must be separated by some computer response to help ensure user attention.

REFERENCE: BB 5.6; EG 4.2.8; MS 5.15.7.4, 5.15.7.5.b.

SEE ALSO: 3.5•7, 4.3•18.

Reversible Control Actions (UNDO) •21

Allow users to UNDO an immediately preceding control action that may have caused an unintended data loss.

COMMENT: Some sort of an UNDO capability is now commonly provided in interface design. UNDO represents one more level of data protection, when warning messages and confirmation procedures fail to prevent error, but can only help the user who notices that an error has been made.

COMMENT: In order to implement an UNDO capability, the computer must maintain a record of data changes resulting from current transactions. How long should that record be, i.e., how many transactions should be reversible? Should a user be able to reverse all transactions back to the beginning of a work session? Or all transactions within some defined sequence? Or just the most recent transaction, as recommended here? Whatever UNDO capability is provided, its limitations should be made clear to users.

COMMENT: Some designers recommend that UNDO itself should be reversible, so that a second UNDO action will do again whatever was just undone. Such a capability implies that UNDO will affect only the last previous transaction, whatever that was. An alternative would be to offer two different UNDO options, one to reverse mistaken actions and one to reverse mistaken UNDO's, risking considerable user confusion.

REFERENCE: MS 5.15.7.7; Lee and Lochovsky, 1983; Nickerson and Pew, 1971; Shneiderman, 1982.

SEE ALSO: 1.3•33, 3.5•10.

6.1 User Identification

User identification procedures should be as simple as possible, consistent with adequate data protection.

•1

Easy LOG-ON

Design the LOG-ON process and procedures for user identification to be as simple as possible consistent with protecting data from unauthorized use.

COMMENT: Some security experts recommend that LOG-ON be made deliberately difficult in order to discourage intruders to a system, and even that the system **not** indicate the successful completion of a LOG-ON sequence. Such measures will confound legitimate users more often than they will impede intruders, and are not recommended here. A better approach would be to keep the initial LOG-ON simple, and then impose some auxiliary procedure to authenticate user identity.

COMMENT: Authentication of user identity is generally **not** enhanced by requiring a user to enter routine data such as terminal, telephone, office or project numbers. In most organizations, those data can readily be obtained by other people. If verification of those data is needed, the user should be asked to review and confirm currently stored values in a supplementary procedure following LOG-ON.

REFERENCE: MS 5.15.7.5.f; Haskett, 1984.

SEE ALSO: 1.8•7.

•2

▸ Prompting LOG-ON

Design the LOG-ON process to provide prompts for all user entries, including passwords and/or whatever other data are required to confirm user identity and to authorize appropriate data access/change privileges.

REFERENCE: EG 4.2.11.

User Choice of Passwords •3

When passwords are required, allow users to choose their own passwords.

COMMENT: A password chosen by a user will generally be easier for that individual to remember. User choice is especially helpful when passwords must be periodically changed, with changing demands on memory.

COMMENT: In the interests of security, users should be given some guidelines in password selection, so that they will not choose easily guessable nicknames or initials, and will choose passwords of sufficient length to resist random guessing.

COMMENT: Some security experts recommend that passwords of nonsense material be composed by a computer and arbitrarily assigned to users, in order to make it more difficult for intruders to guess a password. Such measures will confound legitimate users more often than they will impede intruders, and are not recommended here. A better approach would be to keep passwords memorable, for initial system access, and then impose some auxiliary procedure to authenticate user identity in applications where passwords are considered insufficient protection.

REFERENCE: CSC-STD-002-85; Haskett, 1984.

▶ Changing Passwords •4

Allow users to change their passwords whenever they choose.

COMMENT: A user may sometimes suspect that a password has been disclosed, and thus wish to change it.

COMMENT: In addition to optional changes by users, it may also be good security practice for a system to enforce password changes for all users at periodic intervals.

REFERENCE: CSC-STD-002-85.

▶ Private Entry of Passwords •5

When a password must be entered by a user, ensure that password entry can be private; password entries should **not** be displayed.

COMMENT: Covert entry of passwords will prevent casual eavesdropping by onlookers. This represents an exception to the general recommendation that all entries **should** be displayed.

COMMENT: In the interests of security, it might be noted that passwords should also **not** be retained in readable form in computer memory, although this is not an issue of user interface design.

REFERENCE: MS 5.15.2.2.3; CSC-STD-002-85.

SEE ALSO: 1.0•3.

6.1 User Identification

•6 Limiting Unsuccessful LOG-ON Attempts

Impose a maximum limit on the number and rate of unsuccessful LOG-ON attempts that will provide a margin for user error while protecting the system from persistent attempts at illegitimate access.

COMMENT: Legitimate users will sometimes have difficulty completing a successful LOG-ON, perhaps due to inattention, or a faulty terminal, or faulty communications. Occasional LOG-ON failures of that kind should be tolerable to the system, with the user simply invited to try again.

COMMENT: A record of continuing failure by any particular user to complete successful LOG-ON procedures, including password entry and other tests of claimed user identity, may indicate persistent intrusion attempts. Repeated LOG-ON failures might thus be grounds for denying access to that user. Access might be denied temporarily for some computer-imposed time interval, or indefinitely pending review by a system administrator. The occasional inconvenience to a legitimate user may be tolerable in the interests of increased system security. Analysis of this tradeoff between convenience and security can determine the number and rate of LOG-ON failures that will be tolerated in any particular system application.

REFERENCE: CSC-STD-002-85.

SEE ALSO: 6.0•2.

•7 Auxiliary Tests to Authenticate User Identity

When system security requires more stringent user identification than is provided by password entry, devise auxiliary tests that can authenticate user identity without imposing impractical demands on users' memory.

COMMENT: Various means have been proposed for authenticating user identity, including the use of secret algorithms known only to each individual user. If computer-generated cues and user responses can be protected cryptographically from eavesdropping, a practical scheme might be to require a user to respond to a word association test individually devised by that user for this purpose.

REFERENCE: Haskett, 1984.

Continuous Recognition of User Identity •8

Once a user's identity has been authenticated, ensure that whatever data access/change privileges are authorized for that user will continue throughout a work session.

EXCEPTION: In special instances a user's data access/change privileges might reasonably change as a result of succeeding transactions, e.g., if computer analysis indicated suspicious or otherwise abnormal behavior.

EXCEPTION: A user might reasonably be required to repeat procedures for authentication of identity when resuming work after some specified period of inactivity.

COMMENT: If an identified user is required to take separate actions to authenticate data handling transactions, such as accessing particularly sensitive files or issuing particular commands, the efficiency of system operations may be degraded. Where continuous verification of user identity seems required for data protection, perhaps some automatic means of identification might be devised for that purpose.

REFERENCE: Symons and Schweitzer, 1984.

SEE ALSO: 3.3•10, 6.2•1, 6.2•6, 6.3•1.

6.2 Data Access

> *Data access constraints established to exclude unauthorized users should not hinder legitimate use of data.*

•1 **Single Authorization for Data Access**

Establish user authorization for data access at initial LOG-ON; do **not** require further authentication when a user requests display of particular data.

SEE ALSO: 6.1•8.

•2 **Displayed Security Classification**

When displayed data are classified for security purposes, include a prominent indication of security classification in each display.

COMMENT: This practice will serve to remind users of the need to protect classified data, both in access to the display itself and in any further dissemination of displayed data.

COMMENT: In applications where either real or simulated data can be displayed, a clear indication of simulated data should be included as part of the classification label.

COMMENT: Where a display includes partitioned "windows" of data from different sources, it may be necessary to label security classification separately for each window. Under those conditions, some form of auxiliary coding (e.g., color coding) might help users distinguish a window which contains data at a high security level.

SEE ALSO: 6.0•6.

•3 **Protecting Displayed Data**

When protection of displayed data is essential, maintain computer control over the display and do **not** permit a user to change such "read-only" data.

COMMENT: It is not enough simply to instruct users not to make changes in displayed data. Users may attempt unwanted changes by mistake, or for curiosity, or perhaps even to subvert the system.

REFERENCE: EG 3.4.8; MS 5.15.4.3.12.

SEE ALSO: 2.0•10, 6.3•2.

▶ Indicating "Read-Only" Displays •4

When users are not authorized to change displayed data, indicate that "read-only" status on the display.

COMMENT: In applications where the use of read-only displays is common, then some simple cue in the display header may suffice to indicate that status. In applications where users can usually make additions and/or corrections to displayed data, then any exception to that practice may confuse a user and so should be noted more prominently on the display.

SEE ALSO: 2.0•9.

Protecting Display Formats •5

Protect display formatting features, such as field labels and delimiters, from accidental change by users.

COMMENT: In many data entry tasks users will be allowed to change data fields but should be prevented from making any structural changes to the display. In applications where a user may have to create or modify display formats, special control actions should be provided for that purpose.

REFERENCE: BB 1.8.13; MS 5.15.3.1.1.c.

SEE ALSO: 1.1•23, 1.4•7.

Display Suppression for Security •6

When confidential information is displayed at a work station that might be viewed by casual onlookers, provide the user with some rapid means of temporarily suppressing a current display if its privacy is threatened, and then resuming work later.

COMMENT: Such a capability is sometimes called a "security pause". For quick display suppression a function key might be provided. To retrieve a suppressed display and resume work, a user might be required to make a code entry such as a password, in the interests of data protection.

COMMENT: A suppressed display should not be entirely blank, but should contain an appropriate message indicating its current status, e.g.,

```
Display is temporarily suppressed;
enter password to resume work.
```

SEE ALSO: 3.3•8, 4.2•1, 6.1•8.

6.2 Data Access

•7 **Protecting Printed Data**

As required for security, establish procedures to control access to printed data, rather than simply prohibiting the printing of sensitive data.

COMMENT: User requirements for printed data are often unpredictable, and printing restrictions may handicap task performance. Rather than restrict printing, establish appropriate procedures for restricting further distribution of data printouts.

REFERENCE: BB 4.4.6; EG 4.2.14; MS 5.15.9.2.

SEE ALSO: 2.7.1•11, 5.3•13, 6.4•7.

•8 **Automatic Records of Data Access**

When records of data access are necessary, the computer should keep those records automatically; do **not** rely on users to take critical record keeping actions.

COMMENT: Even cooperative, well-intentioned users can forget to keep manual logs of data access, and will resent the time and effort required to keep such logs. Subversive users, of course, cannot be expected to provide accurate records.

SEE ALSO: 4.5•4.

•9 **Encryption**

When sensitive data may be exposed to unauthorized access, provide a capability for encrypting those data.

COMMENT: Potential exposure may be assumed during any external data transmission, with encryption imposed routinely by the computer. For protection of data within a shared system, a user might choose to encrypt private files to prevent their reading by other people. In such a case, the user must specify a private encryption "key", which will then serve as the basis for automatic encryption by the computer.

SEE ALSO: 6.4•3.

•10 ▶ **Ensuring Reversible Encryption**

Ensure that data encryption is reversible, i.e., that encrypted data are protected from any change that might prevent successful reversal of their encryption.

SEE ALSO: 6.3•2.

Data entry constraints may be needed to prevent unauthorized data change as well as data loss from user errors.

Single Authorization for Data Entry/Change •1

Establish user authorization for data entry/change at initial LOG-ON; do **not** require further authorization when a user attempts particular data entry/change transactions.

SEE ALSO: 6.1•8.

Protection from Data Change •2

When data must **not** be changed, maintain computer control over the data, and do not permit users to change controlled items.

COMMENT: It is not enough simply to instruct users not to make changes in displayed data. Users may attempt unwanted changes by mistake, or for curiosity, or perhaps even to subvert the system.

REFERENCE: MS 5.15.4.3.12.

SEE ALSO: 1.1•23, 1.4•7, 2.0•10, 6.2•3.

Data Entry/Change Transaction Records •3

In situations where unauthorized data changes may be possible, allow users (or a system administrator) to request a record of data entry/change transactions.

COMMENT: Transaction records might be maintained for purposes of user guidance as well as for data protection, as recommended elsewhere.

SEE ALSO: 3.4•3, 4.4•22, 4.5•3.

Simple Procedures •4

Make procedures for data entry/change as simple as possible by following guidelines for the design of data entry functions.

EXAMPLE: Allow users to enter short rather than long items, do not require users to enter leading zeros or count blanks, etc.

COMMENT: Simple procedures will help ensure accuracy in data entry/change transactions.

SEE ALSO: Section 1.

6.3 Data Entry/Change

•5 **Explicit User Actions**

Require users to take some explicit ENTER action to accomplish data entry/change transactions; data change should **not** occur as a possibly unrecognized side effect of other actions.

EXAMPLE: A user should be able to key data into a displayed form but not change stored data unless some explicit ENTER action is taken.

EXAMPLE: A user should be able to point with a lightpen at a displayed item but not change that item unless some further action is taken.

EXCEPTION: In some applications it will be desirable to provide automatic cross-file updating of changed data, or generation of routine, redundant or derived data, without requiring explicit action by a user.

COMMENT: Explicit actions will help direct user attention to data entry/change, and reduce the likelihood of thoughtless errors.

REFERENCE: MS 5.15.2.1.4.

SEE ALSO: 1.0•9, 1.1•4, 3.0•5, 3.1.3•6, 4.0•2, 6.0•9.

•6 ▶ **Single Entry of Related Data**

Allow users to enter logically related items, as in a form-filling dialogue, with a single, explicit action at the end of the sequence, rather than entering each item separately.

COMMENT: This practice permits user review and possible data correction prior to entry. It will also permit efficient cross validation of related data items by the computer.

SEE ALSO: 1.4•1, 6.3•18.

•7 ▶ **Data Entry Independent of Cursor Placement**

Ensure that an ENTER action for multiple data items will result in entry of all items, regardless of where the cursor is placed on the display.

COMMENT: A user may choose to move the cursor back in order to correct earlier data items, and cannot be relied upon to move the cursor forward again. The computer should ignore cursor placement in such cases.

SEE ALSO: 1.1•24.

Editing Data Before Entry •8

Allow users to correct keyed data entries (and control entries) before taking an explicit ENTER action.

COMMENT: Easy correction before entry will avoid the need for computer processing of user-detected errors.

COMMENT: A user should be able to backspace with a nondestructive cursor to the point of error, correct the erroneous item, and ENTER all items without any further cursor positioning.

REFERENCE: EG 5.4; Neal and Emmons, 1984.

SEE ALSO: 1.4•2, 3.5•2, 6.0•10.

Immediate Error Correction •9

When a data entry error is detected by the computer, allow the user to make an immediate correction.

COMMENT: Immediate corrections will be made more easily and accurately. Intended entries will still be fresh in the user's mind, and any source data (e.g., documents) will still be available to the user.

REFERENCE: EG 5.7; MS 5.15.7.7.

SEE ALSO: 1.7•6, 3.5•12.

▶ Editing Entries After Error Detection •10

Following error detection, allow users to edit entries so that they must rekey only those portions that were in error.

COMMENT: If a user must re-enter an entire data set to correct one wrong item, s/he may make new errors in previously correct items.

REFERENCE: BB 5.2.1; EG 4.2.3, 5.4; MS 5.15.7.1.

SEE ALSO: 4.3•15, 6.0•10.

▶ Explicit Entry of Corrections •11

Require users to take an explicit ENTER action for computer processing of error corrections; this should be the same action that was taken to enter the data originally.

REFERENCE: MS 5.15.7.9; PR 4.12.6.

SEE ALSO: 3.5•6, 6.0•9.

6.3 Data Entry/Change

•12 **Flexible BACKUP for Error Correction**

Allow users to return easily to previous steps in a transaction sequence in order to correct an error or make any other desired change.

EXAMPLE: A user might wish to BACKUP through the defined sequence of a question-and-answer dialogue in order to change a previous answer.

COMMENT: The effective implementation of such a BACKUP capability depends upon whether sequences of related transactions can in fact be defined. Any attempt to BACKUP through an arbitrary series of unrelated transactions will pose logical problems both for designers and users.

REFERENCE: MS 5.15.7.7.

SEE ALSO: 3.5•13.

•13 **Data Verification by User Review**

When verification of prior data entries is required, allow users to review and confirm the data, rather than requiring re-entry of the data.

COMMENT: For routine verification, data review by the user will be quicker than re-entry, with less risk of introducing new errors.

COMMENT: For special verification, as when computer processing has detected doubtful and/or discrepant data entries, the user should be alerted with an appropriate advisory message.

SEE ALSO: 1.0•1, 1.8•9, and Section 4.3.

•14 ▶ **Automatic Data Generation**

When routine or redundant data can be derived by the computer, display those data automatically for user review, rather than requiring entry by the user.

COMMENT: This represents an exception, in the interests of improved data accuracy, to the general recommendation that data entry/change should occur only as a result of explicit user actions. Automatic data generation by the computer, where it can be based on derived values or cross-file updating, will be faster and more accurate than user entry. In effect, having a computer do automatically what the user may do poorly is here regarded as a form of data protection.

REFERENCE: BB 2.4.2; MS 5.15.2.1.6.

SEE ALSO: 1.8•7, 1.8•8, 1.8•10, 1.8•11.

▶ Displaying Default Values •15

Display currently operative default values for data entry, so that users can review and confirm them for computer processing.

REFERENCE: BB 2.1.10.

SEE ALSO: 1.8•4, 1.8•5.

Displaying Data to be Changed •16

If a user requests change (or deletion) of a stored data item that is not currently being displayed, offer to display both the old and new values so that the user can confirm or nullify the change before the transaction is completed.

COMMENT: This practice will tend to prevent inadvertent change, including changes resulting in loss of needed data. User attempts at selective data change without displayed feedback will be prone to error.

COMMENT: For proposed deletion of significant amounts of data, such as entire files, it will probably not be feasible to display **all** of the data. In such instances, sufficient information should be provided so that the user can identify those files s/he has selected for deletion. The user should be clearly warned of the potential data loss and required to confirm the destructive action before it will be executed.

SEE ALSO: 1.0•14.

Validating Data Changes •17

Provide data validation software which will detect erroneous or doubtful values for data changes, as well as for initial data entries.

COMMENT: Do not rely on users always to be correct when entering or changing data. When validity of data entries can be checked automatically, such computer validation will improve the accuracy of data entry.

REFERENCE: MS 5.15.2.1.5, 5.15.7.3; PR 4.12.4.

SEE ALSO: 1.7•1.

6.3 Data Entry/Change

•18 ▶ **Cross Validation of Related Data**

For the entry of related data items, provide automatic cross validation to ensure that the data set is logically consistent.

COMMENT: Such cross checking is a significant advantage of on-line data processing, providing computer aids to help users detect logical errors.

REFERENCE: MS 5.15.7.3; PR 4.12.5.

SEE ALSO: 1.4•1, 1.7•1, 6.3•6.

•19 **User Confirmation of Destructive Actions**

Require users to take explicit action to confirm doubtful and/or potentially destructive data change actions before they are accepted by the computer for execution.

COMMENT: A requirement to take an explicit CONFIRM action will direct user attention to questionable data changes, and help the user avoid the consequences of thoughtless errors.

REFERENCE: BB 5.6; MS 5.15.7.5.b.

SEE ALSO: 1.3•32, 1.3•34, 4.3•18, 6.0•18.

•20 **Distinctive File Names**

When data files may be deleted (or overwritten) by name, ensure that the names of different files are distinctive.

COMMENT: If file names are similar, it is easy for users to make an error in file storage, by specifying an unintended overwriting of one file with data from a similarly named other file.

COMMENT: If two or more files are assigned similar names, the distinctive feature should be near the beginning of those names rather than at the end; in particular, no file name should simply be a truncated version of another.

COMMENT: In many applications, file naming is a user option, and distinctive naming will depend on user judgment. In such circumstances, users should be offered guidance on good naming procedures. In addition, perhaps the computer might provide an advisory message if a proposed new file name is similar (e.g., identical in the first 5 letters) to the name of an existing file.

Segregating Real from Simulated Data •21

When simulated data are stored and processed in a system (perhaps for user training), ensure that changes to simulated data are processed separately and do not affect real data.

REFERENCE: BB 6.4.

SEE ALSO: 4.4•30, 6.0•6.

Preventing Data Loss at LOG-OFF •22

When a user requests LOG-OFF, check any pending transactions involving data entry/change and, if data loss seems probable, display an appropriate advisory message to the user.

EXAMPLE:

```
Current data entries have not been filed;
save if needed before confirming LOG-OFF.
```

COMMENT: The user may sometimes suppose that a job is done before taking a necessary further implementing action.

REFERENCE: BB 4.8; MS 5.15.7.5.e.

SEE ALSO: 3.5•11.

6.4 Data Transmission

Data transmission procedures should ensure data protection when sending and receiving messages.

•1 **Automatic Protection of Transmitted Data**

Ensure that whatever measures are adopted to protect data during transmission — e.g., encryption, parity checks, buffering until acknowledgment of receipt — will be applied automatically, without the need for user action.

COMMENT: Users are fallible, and cannot be relied upon to participate quickly and accurately in the mechanisms of data transmission, whereas that is what computers can do well. The computer should check transmitted data to determine whether an error has occurred; correct errors automatically, if necessary by requesting retransmission; and call to the user's attention any case in which a detected error cannot be automatically corrected.

SEE ALSO: 5.4•1, 6.0•1.

•2 **User Review of Data Before Transmission**

When human judgment may be required to determine whether data are appropriate for transmission, provide users (or a system administrator) some means to review outgoing messages and confirm their release before transmission.

COMMENT: Sometimes message release may require coordination among several reviewers in the interests of data protection.

SEE ALSO: 5.3•2.

•3 **Encrypting Messages**

When it is necessary to transmit sensitive data over insecure communication channels, provide automatic encryption to protect such data.

COMMENT: Do not rely on users to remember to request message encryption. A user might be asked to supply an encryption key, but the computer should handle any actual encryption process.

REFERENCE: Price, 1981.

SEE ALSO: 6.0•1, 6.2•9.

Saving Transmitted Data Until Receipt is Confirmed •4

Save a copy of any transmitted message automatically until correct receipt has been confirmed (and possibly longer in some applications).

COMMENT: The primary objective is to prevent irretrievable data loss during transmission. For many system applications, however, the originator of a message will probably want to retain a copy in any case. Any subsequent deletion of that copy should probably be handled as a separate transaction, distinct from data transmission.

Nondisruptive Notification of Messages Received •5

Provide automatic queuing of incoming messages as necessary to ensure that they will not disrupt current user information handling tasks.

COMMENT: In general, incoming data should not replace currently stored data directly, but should be queued for review and disposition by a user. An exception must be made, however, in applications where automatic updating of current situation data is required for operations monitoring, as in air traffic control systems. In such cases data updating is the primary purpose of the system, and that updating should not require continuous actions by a user.

SEE ALSO: 5.5•5.

Authenticating Message Sources •6

When a user must confirm the identity of a message source, provide computer aids for that purpose.

EXAMPLE: In military message systems, received commands might be authenticated automatically by requesting computer-generated confirmation codes.

REFERENCE: Price, 1981.

SEE ALSO: 5.3•6.

Printing Messages •7

Within the constraints of data security, allow users to generate printed copies of transmitted data, including messages sent and received.

COMMENT: User requirements for printed data are often unpredictable, and printing restrictions may handicap task performance. Rather than restrict printing, establish appropriate procedures for restricting further distribution of printed messages.

SEE ALSO: 2.7.1•11, 5.3•13, 6.2•7.

6.5 Design Change

Design change of software supporting data protection may be needed to meet changing operational requirements.

•1 **Flexible Design for Data Protection**

When data protection requirements may change, which is often the case, provide some means for a system administrator to make necessary changes to data protection functions.

COMMENT: Data protection functions that may need to be changed include those represented in these guidelines, namely, changes in protective measures regulating user identification, data access, data entry/change, and data transmission.

•2 **Protection from Design Change**

Protect user interface design from any changes that might impair functions supporting data entry, data display, sequence control, user guidance, data transmission, and data protection.

COMMENT: A trade-off is required between design **flexibility**, to permit needed improvements to the user interface, and design **control**, to protect current functions from undesirable changes. Some form of continuing configuration management should be instituted to evaluate changes to user interface design, just as for any other critical system interface.

SEE ALSO: 1.9•1, 2.8•1, 3.7•1, 4.6•1, 5.6•1.

REFERENCES

Anyone involved in compilation of design guidelines must begin and end by acknowledging the significant contributions of other people. No one person, no matter how wise, can know everything about the complexities of user interface design. Nor will any one person have the perfect judgment and find the perfect words to express that knowledge to an interface designer. Thus when we propose guidelines we must build upon the work of others.

That is a good thing. All design guidelines are necessarily based in some degree on judgment. Thus guidelines development must properly be a collaborative effort. The collective judgment of many people will often prove sounder than the ideas of just one person.

When many people contribute to guidelines development, we must find ways to acknowledge that contribution. One way is to cite previously published papers that pertain to the guidelines. Citations in this report are represented in the reference list that follows. But in the next several pages we also try to acknowledge more direct contributions to our work.

Many of the user interface design guidelines proposed in this report were not invented here, but derive from the ideas of other people. Where the idea for a guideline came from a particular source, an appropriate reference citation has been included for that guideline. Such citation offers credit where credit is due. More importantly, cited references may permit a reader who questions a particular guideline to explore its antecedents, perhaps to gain a better understanding of what is intended.

Citing an external reference in connection with a guideline does not necessarily mean that there is convincing data to support a guideline. Although the references cited here all contain worth-while ideas, only some of these references report results from systematic data collection.

Furthermore, citation of references does **not** necessarily mean that their authors would agree with the wording of guidelines presented here. In some instances, an idea has been borrowed intact. In many more instances, however, ideas have been modified, sometimes drastically, perhaps beyond the intent of their original authors.

In this report, in both the text and the guidelines, citations of specific references are in conventional form, showing author(s) and publication date. Those references are listed in the pages that follow. The particular format used here for citation and listing of references conforms in most respects to the standard referencing practice recently adopted by the Human Factors Society (1984).

However, four reference sources are used generally throughout the guidelines. Those sources are cited so frequently that they have been indicated simply by initials:

BB = Brown, Brown, Burkleo, Mangelsdorf, Olsen, and Perkins, 1983

EG = Engel and Granda, 1975

MS = MIL-STD-1472C (as revised), 1983

PR = Pew and Rollins, 1975

These four general references share a common characteristic — like this report, they are all collections of design guidelines. None of these four general references provide supporting data for their design recommendations, and they need not be consulted for that purpose. The two early reports (EG and PR) have served as a fertile source of ideas for our current guidelines; where those reports are cited here, it means that their early recommendations are still judged to be correct. The two more recent reports (BB and MS) have drawn heavily from common sources, including previous editions of the guidelines proposed here; where those reports are cited, it means that their authors have made similar recommendations to those presented here.

The 1975 IBM report by Engel and Granda (EG) was the first widely recognized compilation of user interface design guidelines. That report has provided inspiration and has served as a seminal reference for others working in this field.

The 1975 BBN report by Pew and Rollins (PR) represents an admirable attempt to propose design guidelines for one particular system application. Its recommendations, however, can readily be generalized for broader application.

The 1983 report by Lin Brown and his colleagues at Lockheed (BB) is a good example of user interface guidelines developed for use as an in-house design standard, but which have also been made available for public reference.

None of these three reports are distributed by government sources such as the National Technical Information Service. However, these reports may be obtained by direct request from their authors.

MIL-STD-1472C (MS), in its current revision, is the US military standard for human engineering in system design. That standard has been cited here 237 times, for 209 guidelines. It is important to emphasize that guidelines do not carry the same weight as design standards. Guidelines are proposed here for optional application in system development, rather than to be imposed contractually. However, there is some considerable correspondence in content between these guidelines and the current military standard.

Not all ideas for guidelines come from published references. Some of the guidelines proposed here have resulted from discussion with professional colleagues. And the wording of all guidelines has been improved through critical review of earlier published versions. Over the past several years, a number of people have contributed suggestions for improving the guidelines material:

Sara R. Abbott	Union Carbide Corporation
James H. Alexander	Tektronix, Inc.
Dorothy J. Antetomaso	The MITRE Corporation
Christopher J. Arbak	McDonnell Douglas Corporation
Arlene F. Aucella	Wang Laboratories
Clifford E. Baker	The MITRE Corporation
J. David Beattie	Ontario Hydro
Leo Beltracchi	US Nuclear Regulatory Commission
C. Marlin Brown	Lockheed Missiles and Space Company
Alphonse Chapanis	Alphonse Chapanis Ph.D.

Hal Cheney	OCLC
Kent B. Davis	Litton Data Command Systems
Robert S. Didner	Decision Information Designs
John Dinan	Raytheon Equipment Division
Susan M. Dray	Honeywell, Inc.
Joseph S. Dumas	American Institutes for Research
Sam L. Ehrenreich	AT&T Bell Laboratories
Jeanne Fleming	The MITRE Corporation
James D. Foley	The George Washington University
Elaine A. Fournier	The MITRE Corporation
Wilbert O. Galitz	Galitz, Inc.
Robert N. Gifford	Northrop Electronics
Susan R. Gilbert	Wang Laboratories
Nancy C. Goodwin	The MITRE Corporation
Jo Huddleston	Ferranti Computer Systems Limited
Richard M. Kane	Wang Laboratories
Richard S. Keister	Battelle Columbus Laboratories
Karen L. Kessel	Hughes Aircraft Company
Judith R. Kornfeld	Symbolics, Inc.
Jack I. Laveson	Integrated Systems Research
Richard Marshall	Olivetti
Harold Miller-Jacobs	Sperry Corporation
Alice M. Mulvehill	The MITRE Corporation
Jakob Nielsen	Technical University of Denmark
Lorraine F. Normore	Chemical Abstracts Service
Robert N. Parrish	The Aerospace Corporation
Steven P. Rogers	Anacapa Sciences, Inc.
Eric M. Schaffer	Human Performance Associates
Ben Shneiderman	University of Maryland
Malcolm L. Stiefel	The MITRE Corporation
Nancy S. Tanner	University of Massachusetts
John C. Thomas	Nynex Corporation
Herb Weiner	Tektronix, Inc.
R. Don Williams	Texas Instruments, Inc.

Some of these people have offered specific suggestions. Some have contributed more general comments about the wording or formatting of the guidelines material. But all have shown a serious concern with trying to improve the guidelines and make them more useful to designers of user interface software. Probably not one of these people would agree with all of the guidelines proposed here; in matters of judgment we can seldom achieve unanimity. But where the guidelines seem good, these are people who deserve our thanks.

Several people on this list deserve extra thanks. Our colleagues at MITRE have continued to serve as an in-house working group for guidelines review. Special thanks are due to Nancy Goodwin for her thorough revision of the guidelines on data transmission; to Jeanne Fleming and James Foley for their

detailed comments on guidelines proposed for graphics entry and display; and to Dorothy Antetomaso for her review of the guidelines on data security. Special thanks for past contributions are due to Arlene Aucella who helped prepare the 1983 guidelines report; and to MITRE supervisor Marlene Hazle for her early encouragement and support of guidelines compilation.

Albert, A. E. (1982). The effect of graphic input devices on performance in a cursor positioning task. In *Proceedings of the Human Factors Society 26th Annual Meeting* (pp. 54-58). Santa Monica, CA: Human Factors Society.

Aretz, A. J., and Kopala, C. J. (1981). Automatic return in multifunction control logic. In *Proceedings of the Human Factors Society 25th Annual Meeting* (pp. 254-256). Santa Monica, CA: Human Factors Society.

Badre, A. N. (1984). Designing transitionality into the user-computer interface. In G. Salvendy (Ed.), *Human-Computer Interaction* (pp. 27-34). Amsterdam: Elsevier Science Publishers.

Barnard, P. J., Hammond, N. V., MacLean, A., and Morton, J. (1982). Learning and remembering interactive commands in a text-editing task. *Behaviour and Information Technology, 1*, 347-358.

Barnard, P., and Marcel, T. (1984). Representation and understanding in the use of symbols and pictograms. In R. Easterby and H. Zwaga (Eds.), *Information Design* (pp. 37-75). Chichester: Wiley.

Bauer, D. W., and Eddy, J. K. (1986). The representation of command language syntax. *Human Factors, 28*, 1-10.

Bertelson, P., Boons, J.-P., and Renkin, A. (1965). Vitesse libre et vitesse imposee dans une tache simulant le tri mechanique de la correspondence [Self pacing and imposed pacing in a task simulating automated postal sorting]. *Ergonomics, 8*, 3-22.

Bertoni, P. (1982, June 22). Of slipped disks . . . . Boston *Globe*.

Bewley, W. L., Roberts, T. L., Schroit, D., and Verplank, W. L. (1983). Human factors testing in the design of Xerox's 8010 "Star" office workstation. In *Proceedings of CHI'83 Human Factors in Computing Systems* (pp. 72-77). New York: Association for Computing Machinery.

Bigelow, C. (1985, July). Proceedings of the Typography Interest Group ACM CHI'85. *SIGCHI Bulletin, 17*(1), 10-11.

Billingsley, P. A. (1982). Navigation through hierarchical menu structures: Does it help to have a map? In *Proceedings of the Human Factors Society 26th Annual Meeting* (pp. 103-107). Santa Monica, CA: Human Factors Society.

BB = Brown, C. M., Brown, D. B., Burkleo, H. V., Mangelsdorf, J. E., Olsen, R. A., and Perkins, R. D. (1983, June 15). *Human Factors Engineering Standards for Information Processing Systems* (LMSC-D877141). Sunnyvale, CA: Lockheed Missiles and Space Company.

Bruder, J., Moy, M., Mueller, A., and Danielson, R. (1981). User experience and evolving design in a local electronic mail system. In R. P. Uhlig (Ed.), *Computer Message Systems* (pp. 69-78). New York: North-Holland.

Bury, K. F., Boyle, J. M., Evey, R. J., and Neal, A. S. (1982). Windowing versus scrolling on a visual display terminal. *Human Factors, 24*, 385-394.

Butterbaugh, L. C., and Rockwell, T. H. (1982). Evaluation of alternative alphanumeric keying logics. *Human Factors, 24*, 521-533.

Campbell, A. J., Marchetti, F. M., and Mewhort, D. J. K. (1981). Reading speed and text production: A note on right-justification techniques. *Ergonomics, 24*, 633-640.

Carroll, J. M. (1982). Learning, using and designing filenames and command paradigms. *Behaviour and Information Technology, 1*, 327-346.

Chao, B. P. (1985). Evaluation of a video storage system for intrusion detections. In *Proceedings of the Human Factors Society 29th Annual Meeting* (pp. 417-421). Santa Monica, CA: Human Factors Society.

Cleveland, W. S. (1985). *The Elements of Graphing Data.* Monterey, CA: Wadsworth Advanced Books and Software.

Cohill, A. M., and Williges, R. C. (1985). Retrieval of HELP information for novice users of interactive computer systems. *Human Factors, 27*, 335-343.

CSC-STD-002-85. (1985, 12 April). *Department of Defense Password Management Guideline.* Fort George G. Meade, MD: Department of Defense Computer Security Center.

Dean, M. (1982). How a computer should talk to people. *IBM Systems Journal, 21*, 424-453.

Demers, R. A. (1981). System design for usability. *Communications of the ACM, 24*, 494-501.

Deutsch, D. (1981). Design of a message format standard. In R. P. Uhlig (Ed.), *Computer Message Systems* (pp. 199-220). New York: North-Holland.

Deutsch, D. P. (1984). Implementing distribution lists in computer-based message systems. In H. T. Smith (Ed.), *Computer-Based Message Services* (pp. 3-13). New York: North-Holland.

Dray, S. M., Ogden, W. G., and Vestewig, R. E. (1981). Measuring performance with a menu-selection human-computer interface. In *Proceedings of the Human Factors Society 25th Annual Meeting* (pp. 746-748). Santa Monica, CA: Human Factors Society.

Duchnicky, R. L., and Kolers, P. A. (1983). Readability of text scrolled on visual display terminals as a function of window size. *Human Factors, 25,* 683-692.

Dunn, C. (1973). The use of real-time simulation by means of animation film as an analytical design tool in certain spatio-temporal situations. *Ergonomics, 16,* 515-519.

Durding, B. M., Becker, C. A., and Gould, J. D. (1977). Data organization. *Human Factors, 19,* 1-14.

Dwyer, B. (1981). A user-friendly algorithm. *Communications of the ACM, 24,* 556-561.

Dzida, W. (1981). Computer mediated messaging for interactive purposes. In R. P. Uhlig (Ed.), *Computer Message Systems* (pp. 79-87). New York: North-Holland.

Ehrenreich, S. L. (1981). Query languages: Design recommendations derived from the human factors literature. *Human Factors, 23,* 709-725.

Ehrenreich, S. L. (1985). Computer abbreviations: Evidence and synthesis. *Human Factors, 27,* 143-156.

Ehrenreich, S. L., and Porcu, T. (1982). Abbreviations for automated systems: Teaching operators the rules. In A. Badre and B. Shneiderman (Eds.), *Directions in Human/Computer Interaction* (pp. 111-135). Norwood, NJ: Ablex Publishing.

Elkerton, J., Williges, R. C., Pittman, J. A., and Roach, J. (1982). Strategies of interactive file search. In *Proceedings of the Human Factors Society 26th Annual Meeting* (pp. 83-86). Santa Monica, CA: Human Factors Society.

EG = Engel, S. E., and Granda, R. E. (1975, December). *Guidelines for Man/Display Interfaces* (Technical Report TR 00.2720). Poughkeepsie, NY: IBM.

Foley, J. D., and Van Dam, A. (1982). *Fundamentals of Interactive Computer Graphics.* Reading, MA: Addison-Wesley.

Foley, J. D., and Wallace, V. L. (1974). The art of natural graphic man-machine conversation. *Proceedings of the IEEE, 62,* 462-471.

Foley, J. D., Wallace, V. L., and Chan, P. (1984, November). The human factors of computer graphics interaction techniques. *IEEE Computer Graphics and Applications, 4*(11), 13-48.

Gade, P. A., Fields, A. F., Maisano, R. E., Marshall, C. F., and Alderman, I. N. (1981). Data entry performance as a function of method and instructional strategy. *Human Factors, 23,* 199-210.

Galitz, W. O. (1980). *Human Factors in Office Automation.* Atlanta, GA: Life Office Management Association.

Garcia-Luna, J. J., and Kuo, F. L. (1981). Addressing and directory systems for large computer mail systems. In R. P. Uhlig (Ed.), *Computer Message Systems* (pp. 297-313). New York: North-Holland.

Gardan Y., and Lucas, M. (1984). *Interactive Graphics in CAD.* London: Kogan Page.

Geer, C. W. (1981). *Human Engineering Procedures Guide* (Technical Report AFAMRL-TR-81-35). Wright-Patterson Air Force Base, OH: Air Force Aerospace Medical Research Laboratory. (NTIS No. AD A108 643)

Geiser, G., Schumacher, W., and Berger, L. (1982). Talking keyboard for user guidance in multifunction systems. In *Proceedings of the Human Factors Society 26th Annual Meeting* (pp. 436-439). Santa Monica, CA: Human Factors Society.

Gilfoil, D. M. (1982). Warming up to computers: A study of cognitive and affective interaction over time. In *Proceedings of Conference on Human Factors in Computer Systems* (pp. 245-250). Washington, DC: Association for Computing Machinery.

Good, M. D., Whiteside, J. A., Wixon, D. R., and Jones, S. J. (1984). Building a user-derived interface. *Communications of the ACM, 27,* 1032-1043.

Goodwin, N. C. (1974, March). *INTRO — In Which a Smart Terminal Teaches Its Own Use* (Technical Report ESD-TR-74-374). Hanscom Air Force Base, MA: USAF Electronic Systems Division. (NTIS No. AD A126 205)

Goodwin, N. C. (1975). Cursor positioning on an electronic display using lightpen, lightgun, or keyboard for three basic tasks. *Human Factors, 17,* 289-295.

Goodwin, N. C. (1980). A user-oriented evaluation of computer-aided message handling. In *Proceedings of the Human Factors Society 24th Annual Meeting* (pp. 585-589). Santa Monica, CA: Human Factors Society.

Goodwin, N. C. (1982). Effect of interface design on usability of message handling systems. In *Proceedings of the Human Factors Society 26th Annual Meeting* (pp. 69-73). Santa Monica, CA: Human Factors Society.

Goodwin, N. C. (1983). Designing a multipurpose menu driven user interface to computer based tools. In *Proceedings of the Human Factors Society 27th Annual Meeting* (pp. 816-820). Santa Monica, CA: Human Factors Society.

Goodwin, N. C. (1984). Building a usable office support system from diverse components. In B. Shackel (Ed.), *Human-Computer Interaction INTERACT'84* (pp. 577-581). Amsterdam: Elsevier Science Publishers.

Gould, J. D. (1981). Composing letters with computer-based text editors. *Human Factors, 23*, 593-606.

Gould, J. D., and Grischkowsky, N. (1984). Doing the same work with hard copy and with cathode-ray tube (CRT) computer terminals. *Human Factors, 26*, 323-337.

Gould, J. D., and Lewis, C. (1983). Designing for usability — Key principles and what designers think. In *Proceedings of CHI'83 Human Factors in Computing Systems* (pp. 50-53). New York: Association for Computing Machinery.

Granda, R. E., Teitelbaum, R. C., and Dunlap, G. L. (1982). The effect of VDT command line location on data entry behavior. In *Proceedings of the Human Factors Society 26th Annual Meeting* (pp. 621-624). Santa Monica, CA: Human Factors Society.

Gregory, M., and Poulton, E. C. (1970). Even versus uneven right-hand margins and the rate of comprehension in reading. *Ergonomics, 13*, 427-434.

Grudin, J., and Barnard, P. (1984). The cognitive demands of learning and representing command names for text editing. *Human Factors, 26*, 407-422.

Hakkinen, M. T., and Williges, B. H. (1984). Synthesized warning messages: Effects of an alerting cue in single- and multiple-function voice synthesis systems. *Human Factors, 26*, 185-195.

Hamill, B. W. (1980). Experimental document design: Guidebook organization and index formats. In *Proceedings of the Human Factors Society 24th Annual Meeting* (pp. 480-483). Santa Monica, CA: Human Factors Society.

Hannemyr, G., and Innocent, P. R. (1985). A network user interface: Incorporating human factors guidelines into the ISO standard for open systems interconnection. *Behaviour and Information Technology, 4*, 309-326.

Hanson, R. H., Payne, D. G., Shiveley, R. J., and Kantowitz, B. H. (1981). Process control simulation research in monitoring analog and digital displays. In *Proceedings of the Human Factors Society 25th Annual Meeting* (pp. 154-158). Santa Monica, CA: Human Factors Society.

Hartley, J., Young, M., and Burnhill, P. (1975). On the typing of tables. *Applied Ergonomics, 6*, 39-42.

Haskett, J. A. (1984). Pass-algorithms: A user validation scheme based on knowledge of secret algorithms. *Communications of the ACM, 27*, 777-781.

Hayes, P. (1985). A panel on the utility of natural languages. In *Proceedings of CHI'85 Conference on Human Factors in Computing Systems* (p. 19). New York: Association for Computing Machinery.

Hirsh-Pasek, K., Nudelman, S., and Schneider, M. L. (1982). An experimental evaluation of abbreviation schemes in limited lexicons. *Behaviour and Information Technology, 1*, 359-369.

Hollingsworth, S. R., and Dray, S. M. (1981). Implications of post-stimulus cueing of response options for the design of function keyboards. In *Proceedings of the Human Factors Society 25th Annual Meeting* (pp. 263-265). Santa Monica, CA: Human Factors Society.

Hopkin, V. D., and Taylor, R. M. (1979). *Human Factors in the Design and Evaluation of Aviation Displays* (Report AGARD-AG-225). Neuilly sur Seine: NATO Advisory Group for Aerospace Research and Development. (NTIS No. AD A076 631)

Human Factors Society. (1984). *Author's Guide.* Santa Monica, CA: Author.

Keister, R. S., and Gallaway, G. R. (1983). Making software user friendly: An assessment of data entry performance. In *Proceedings of the Human Factors Society 27th Annual Meeting* (pp. 1031-1034). Santa Monica, CA: Human Factors Society.

Koved, L., and Shneiderman, B. (1986). Embedded menus: Selecting items in context. *Communications of the ACM, 29*, 312-318.

Krohn, G. S. (1983). Flowcharts used for procedural instructions. *Human Factors, 25*, 573-581.

Lee, A., and Lochovsky, F. H. (1983). Enhancing the usability of an office information system through direct manipulation. In *Proceedings of CHI'83 Human Factors in Computing Systems* (pp. 130-134). New York: Association for Computing Machinery.

Liebelt, L. S., McDonald, J. E., Stone, J. D., and Karat, J. (1982). The effect of organization on learning menu access. In *Proceedings of the Human Factors Society 26th Annual Meeting* (pp. 546-550). Santa Monica, CA: Human Factors Society.

Limanowski, J. J. (1983). On-line documentation systems: History and issues. In *Proceedings of the Human Factors Society 27th Annual Meeting* (pp. 1027-1030). Santa Monica, CA: Human Factors Society.

Magers, C. S. (1983). An experimental evaluation of on-line HELP for non-programmers. In *Proceedings of CHI '83 Human Factors in Computing Systems* (pp. 277-281). New York: Association for Computing Machinery.

Martin, J. (1973). *Design of Man-Computer Dialogues.* Englewood Cliffs, NJ: Prentice-Hall.

Martin, J. (1978). *The Wired Society.* Englewood Cliffs, NJ: Prentice-Hall.

McDonald, J. E., Stone, J. D., and Liebelt, L. S. (1983). Searching for items in menus: The effects of organization and type of target. In *Proceedings of the Human Factors Society 27th Annual Meeting* (834-837). Santa Monica, CA: Human Factors Society.

Michard, A. (1982). Graphical presentation of boolean expressions in a database query language: Design notes and an ergonomic evaluation. *Behaviour and Information Technology, 1,* 279-288.

MIL-H-48655B. (1979, 31 January). *Military Specification: Human Engineering Requirements for Military Systems, Equipment and Facilities.* Washington, DC: Department of Defense.

MIL-STD-12D. (1981, 29 May). *Abbreviations for Use on Drawings, and in Specifications, Standards and Technical Documents.* Washington, DC: Department of Defense.

MIL-STD-1472B. (1974, 31 December). *Military Standard: Human Engineering Design Criteria for Military Systems, Equipment and Facilities.* Washington, DC: Department of Defense.

MS = MIL-STD-1472C, Revised. (1983, 1 September). *Military Standard: Human Engineering Design Criteria for Military System, Equipment and Facilities.* Washington, DC: Department of Defense.

Miller, D. P. (1981). The depth/breadth tradeoff in hierarchical computer menus. In *Proceedings of the Human Factors Society 25th Annual Meeting* (pp. 296-300). Santa Monica, CA: Human Factors Society.

Miller, R. B. (1968). Response time in user-system conversational transactions. In *Proceedings of the AFIPS Fall Joint Computer Conference, 33,* 267-277.

Milroy, R., and Poulton, E. C. (1978). Labelling graphs for improved reading speed. *Ergonomics, 21*, 55-61.

Mooers, C. D. (1983). Changes that users demanded in the user interface to the Hermes message system. In *Proceedings of CHI'83 Human Factors in Computing Systems* (pp. 88-92). New York: Association for Computing Machinery.

Morrill, C. S. (1967). Computer-aided instruction as part of a management information system. *Human Factors, 9*, 251-256.

Morrill, C. S., and Davies, B. L. (1961). Target tracking and acquisition in three dimensions using a two-dimensional display surface. *Journal of Applied Psychology, 45*, 214-221.

Morrill, C. S., Goodwin, N. C., and Smith, S. L. (1968). User input mode and computer-aided instruction. *Human Factors, 10*, 225-232.

Moses, F. L., and Ehrenreich, S. L. (1981). Abbreviations for automated systems. In *Proceedings of the Human Factors Society 25th Annual Meeting* (pp. 132-135). Santa Monica, CA: Human Factors Society.

Mosier, J. N., and Smith, S. L. (1986). Application of guidelines for designing user interface software. *Behaviour and Information Technology, 5*, 39-46.

Muter, P., Latremouille, S. A., Treurniet, W. C., and Beam, P. (1982). Extended readings of continuous text on television screens. *Human Factors, 24*, 501-508.

Muter, P., and Mayson, C. (1986). The role of graphics in item selection from menus. *Behaviour and Information Technology, 5*, 89-95.

Myers, B. A. (1985). The importance of percent-done progress indicators for computer-human interfaces. In *Proceedings of CHI'85 Human Factors in Computing Systems* (pp. 11-17). New York: Association for Computing Machinery.

NASA (National Aeronautics and Space Administration). (1979, January). *Spacelab Experiment Computer Application Software (ECAS) Display Design and Command Usage Guidelines* (Report MSFC-PROC-711). George C. Marshall Space Flight Center, AL: Author.

Neal, A. S., and Darnell, M. J. (1984). Text-editing performance with partial-line, partial-page, and full-page displays. *Human Factors, 26*, 431-441.

Neal, A. S., and Emmons, W. H. (1984). Error correction during text entry with word-processing systems. *Human Factors, 24*, 443-447.

Nickerson, R. S., and Pew, R. W. (1971, June). Oblique steps toward the human-factors engineering of interactive computer systems. Published as an Appendix in M. C. Grignetti, D. C. Miller, R. S. Nickerson and R. W. Pew, *Information Processing Models and Computer Aids for Human Performance* (Report No. 2190). Cambridge, MA: Bolt, Beranek and Newman. (NTIS No. AD A732 913)

Noyes, L. (1980). The positioning of type on maps: The effect of surrounding material on word recognition. *Human Factors, 22*, 353-360.

Pakin, S. E., and Wray, P. (1982). Designing screens for people to use easily. *Data Management, 20*(7), 36-41.

Palme, J. (1979). A human-computer interface for non-computer specialists. *Software - Practice and Experience, 9*, 741-747.

Parsons, H. M. (1970). The scope of human factors in computer-based data processing systems. *Human Factors, 12*, 165-175.

Penniman, W. D. (1979, May). Past chairman's message. *SIG Newsletter No. UOI-10.* Washington, DC: American Society for Information Science.

PR = Pew, R. W., and Rollins, A. M. (1975). *Dialog Specification Procedures* (Report 3129, revised). Cambridge, MA: Bolt Beranek and Newman.

Phillips R. J. (1982). An experimental investigation of layer tints for relief maps in school atlases. *Ergonomics, 25*, 1143-1154.

Poller, M. F., and Garter, S. F. (1984). The effects of modes on text editing by experienced editor users. *Human Factors, 26*, 449-462.

Price, W. L. (1981). Encryption in computer networks and message systems. In R. P. Uhlig (Ed.), *Computer Message Systems* (pp. 413-423). New York: North-Holland.

Radin, D. I. (1984). Effects of command language punctuation on human performance. In G. Salvendy (Ed.), *Human-Computer Interaction* (pp. 177-180). Amsterdam: Elsevier Science Publishers.

Ramsey, H. R., and Atwood, M. E. (1979, September). *Human Factors in Computer Systems: A Review of the Literature* (Technical Report SAI-79-111-DEN). Englewood, CO: Science Applications, Inc. (NTIS No. AD A075 679)

Ramsey, H. R., and Atwood, M. E. (1980). Man-computer interface design guidance: State of the art. In *Proceedings of the Human Factors Society 24th Annual Meeting* (pp. 85-89). Santa Monica, CA: Human Factors Society.

Ramsey, H. R., Atwood, M. E., and Kirshbaum, P. J. (1978, May). *A Critically Annotated Bibliography of the Literature of Human Factors in Computer Systems* (Technical Report SAI-78-070-DEN). Englewood, CO: Science Applications, Inc. (NTIS No. AD A058 081)

Reisner, P. (1977). Use of psychological experimentation as an aid to development of a query language. *IEEE Transactions on Software Engineering, SE-3*, 218-229.

Reisner, P. (1981). Formal grammar and human factors design of an interactive graphics system. *IEEE Transactions on Software Engineering, SE-7*, 229-240.

Roberts, T. L., and Moran, T. P. (1983). The evaluation of text editors: methodology and empirical results. *Communications of the ACM, 26*, 265-283.

Rogers, W. H., and Moeller, G. (1984). Comparison of abbreviation methods: Measures of preference and decoding performance. *Human Factors, 26*, 49-59.

Savage, R. E., Habinek, J. K., and Blackstad, N. J. (1982). An experimental evaluation of input field and cursor combinations. In *Proceedings of the Human Factors Society 26th Annual Meeting* (pp. 629-633). Santa Monica, CA: Human Factors Society.

Schutz, H. G. (1961). An evaluation of formats for graphic trend displays - Experiment II. *Human Factors, 3*, 99-107.

Seibel, R. (1972). Data entry devices and procedures. In H. P. Van Cott and R. G. Kinkade (Eds.), *Human Engineering Guide to Equipment Design* (pp. 311-344). Washington, DC: U. S. Government Printing Office.

Shinar, D., Stern, H. I., Bubis, G., and Ingram, D. (1985). The relative effectiveness of alternative strategies in menu driven computer programs. In *Proceedings of the Human Factors Society 29th Annual Meeting* (pp. 645-649). Santa Monica, CA: Human Factors Society.

Shneiderman, B. (1980). *Software Psychology: Human Factors in Computer and Information Systems.* Cambridge, MA: Winthrop Publishers.

Shneiderman, B. (1981). A note on human factors issues of natural language interaction with database systems. *Information Systems, 6*, 125-129.

Shneiderman, B. (1982). The future of interactive systems and the emergence of direct manipulation. *Behaviour and Information Technology, 1*, 237-256.

Shneiderman, B. (1984). Response time and display rate in human performance with computers. *Computing Surveys, 16*, 265-285.

Sidorsky, R. C., Parrish, R. N., Gates, J. L., and Munger, S. J. (1984, May). *Design guidelines for user transactions with battlefield automated systems: Prototype for a handbook* (ARI Research Product 84-08). Alexandria, VA: US Army Research Institute. (NTIS No. AD A513 231)

Simpson, C. A., McCauley, M. E., Roland, E. F., Ruth, J. C., and Williges, B. H. (1985). System design for speech recognition and generation. *Human Factors, 27*, 115-142.

Simpson, C. A., and Williams, D. H. (1980). Response time effects of alerting tone and semantic context for synthesized voice cockpit warnings. *Human Factors, 22*, 319-330.

Smith, D. C., Irby, C., Kimball, R., and Verplank, B. (1982). Designing the Star user interface. *BYTE, 7*(4), 242-282.

Smith, H. T. (Ed.) (1984). *Computer-Based Message Services.* New York: North-Holland.

Smith, S. L. (1962a). Angular estimation. *Journal of Applied Psychology, 46*, 240-246.

Smith, S. L. (1962b). Color coding and visual search. *Journal of Experimental Psychology, 64*, 434-440.

Smith, S. L. (1963a). Color coding and visual separability in information displays. *Journal of Applied Psychology, 47*, 358-364. (a)

Smith, S. L. (1963b). Man-computer information transfer. In J. H. Howard (Ed.), *Electronic Information Display Systems* (pp. 284-299). Washington, DC: Spartan Books.

Smith, S. L. (1980, June). *Requirements definition and design guidelines for the man-machine interface in C3 system acquisition* (Technical Report ESD-TR-80-122). Hanscom Air Force Base, MA: USAF Electronic Systems Division. (NTIS No. AD A087 258)

Smith, S. L. (1981a, February). *Man-Machine Interface (MMI) Requirements Definition and Design Guidelines: A Progress Report* (Technical Report ESD-TR-81-113). Hanscom Air Force Base, MA: USAF Electronic Systems Division. (NTIS No. AD A096 705)

Smith, S. L. (1981b). Exploring compatibility with words and pictures. *Human Factors, 23*, 305-315.

Smith, S. L. (1982a). "User-system interface". Human Factors Society *Bulletin, 25*(3), 1.

Smith, S. L. (1982b, April). *User-System Interface Design for Computer-Based Information Systems* (Technical Report ESD-TR-82-132). Hanscom Air Force Base, MA: USAF Electronic Systems Division. (NTIS No. AD A115 853)

Smith, S. L. (1986). Standards versus guidelines for designing user interface software. *Behaviour and Information Technology, 5*, 47-61.

Smith, S. L., and Aucella, A. F. (1983a, March). *Design Guidelines for the User Interface to Computer-Based Information Systems* (Technical Report ESD-TR-83-122). Hanscom Air Force Base, MA: USAF Electronic Systems Division. (NTIS No. AD A127 345)

Smith, S. L., and Aucella, A. F. (1983b). Numbering formats for hierarchic lists. *Human Factors, 25*, 343-348.

Smith, S. L., Farquhar, B. B., and Thomas, D. W. (1965). Color coding in formatted displays. *Journal of Applied Psychology, 49*, 393-398.

Smith, S. L., and Goodwin, N. C. (1970). Computer-generated speech and man-computer interaction. *Human Factors, 12*, 215-223.

Smith, S. L., and Goodwin, N. C. (1971a). Alphabetic data entry via the Touch-Tone pad: A comment. *Human Factors, 13*, 189-190.

Smith, S. L., and Goodwin, N. C. (1971b). Blink coding for information display. *Human Factors, 13*, 283-290.

Smith, S. L., and Goodwin, N. C. (1972). Another look at blinking displays. *Human Factors, 14*, 345-347.

Smith, S. L., and Mosier, J. N. (1984a). *Design Guidelines for User-System Interface Software* (Technical Report ESD-TR-84-190). Hanscom Air Force Base, MA: USAF Electronic Systems Division. (NTIS No. AD A154 907)

Smith, S. L., and Mosier, J. N. (1984b). The user interface to computer-based information systems: A survey of current software design practice. *Behaviour and Information Technology, 3*, 195-203.

Smith, S. L., and Thomas, D. W. (1964). Color versus shape coding in information displays. *Journal of Applied Psychology, 48*, 137-146.

Snowberry, K., Parkinson, S. R., and Sisson, N. (1983). Computer display menus. *Ergonomics, 26*, 699-712.

Stewart, T. (1980). Communicating with dialogues. *Ergonomics, 23*, 909-919.

Symons, C. R., and Schweitzer, J. A. (1984). A proposal for an automated access control standard. *ACM/SIGCHI Bulletin, 16*(1), 17-23.

Tammaro, S. G. (1983). An analysis of the use of computer-based office support tools: User characteristics and usage patterns. In *Proceedings of the Human Factors Society 27th Annual Meeting* (pp. 882-886). Santa Monica, CA: Human Factors Society.

Thimbleby, H. (1985). Failure in the technical user-interface design process. *Computers and Graphics, 9*, 187-193.

Thomas, J. C., and Rosson, M. B. (1984). Human factors and synthetic speech. In B. Shackel (Ed.), *Human-Computer Interaction INTERACT'84* (pp. 37-42). Amsterdam: Elsevier Science Publishers.

Thompson, D. A. (1971). Interface design for an interactive information retrieval system: A literature survey and a research system description. *Journal of the American Society for Information Science, 22*, 361-373.

Trollip, S. R., and Sales, G. (1986). Readability of computer-generated fill-justified text. *Human Factors, 28*, 159-163.

Tucker, J. B. (1984, June). Computer graphics achieves new realism. *High Technology*, 40-53.

Tufte, E. R. (1983). *The Visual Display of Quantitative Information*. Cheshire, CT: Graphics Press.

Tullis, T. S. (1981). An evaluation of alphanumeric, graphic, and color information displays. *Human Factors, 23*, 541-550.

Tullis, T. S. (1983). The formatting of alphanumeric displays: A review and analysis. *Human Factors, 25*, 657-682.

Uhlig, R. P. (Ed.) (1981). *Computer Message Systems*. New York: North-Holland.

Weitzman, D. O. (1985). Color coding re-viewed. In *Proceedings of the Human Factors Society 29th Annual Meeting* (pp 1079-1083). Santa Monica, CA: Human Factors Society.

Whalley, P. C., and Fleming, R. W. (1975). An experiment with a simple recorder of reading behaviour. *Programmed Learning and Educational Technology, 12*, 120-124.

Whitfield, D., Ball R. G., and Bird, J. M. (1983). Some comparisons of on-display and off-display touch input devices for interaction with computer generated displays. *Ergonomics, 26*, 1033-1053.

Williamson, H., and Rohlfs, S. (1981). The user interface design process. In R. P. Uhlig (Ed.), *Computer Message Systems* (pp. 427-445). New York: North-Holland.

Williges, R. C. (1984). Design of human-computer dialogues. In G. Salvendy (Ed.), *Human-Computer Interaction* (pp.35-42). Amsterdam: Elsevier Science Publishers.

Woodson, W. E. (1981). *Human Factors Design Handbook.* New York: McGraw-Hill.

Wright, P. (1977). Presenting technical information: A survey of research findings. *Instructional Science, 6,* 93-134.

Wright, P., and Reid, F. (1973). Written information: Some alternatives to prose for expressing the outcomes of complex contingencies. *Journal of Applied Psychology, 57,* 160-166.

Zoltan-Ford, E. (1984). Reducing variability in natural-language interactions with computers. In *Proceedings of the Human Factors Society 28th Annual Meeting* (pp 768-772). Santa Monica, CA: Human Factors Society.

GUIDELINE TITLES

The total bulk of these 944 guidelines is too great for easy scanning. If a reader wants to find guidelines pertaining to a particular subject, simply leafing through this report will probably not help. Some more efficient means is needed to find the guidelines that cover any specific topic.

The report begins with a table of contents, but that shows only the overall functional organization of the guidelines material. Section titles are given, but there is no information about specific topical coverage within sections.

The report ends with a topical index. But any index, no matter how carefully prepared, will sometimes fail to direct us to the material we seek. One problem is that a seeker's words for a particular topic will sometimes not match the words chosen by the person who constructed the index.

Listed here are the titles of all 944 guidelines, in the words actually used. This listing is more detailed than the table of contents but more structured than the index. This listing provides a relatively compact summary of topical coverage within the different sections of the guidelines. A reader who is not already familiar with the guidelines may find these titles helpful for locating specific information.

DATA ENTRY

1.0 General

- •1 Data entered only once
- •2 Entry via primary display
- •3 Feedback during data entry
- •4 ▶ Fast response
- •5 Single method for entering data
- •6 Defined display areas for data entry
- •7 Consistent method for data change
- •8 User-paced data entry
- •9 Explicit ENTER action
- •10 ▶ ENTER key labeling
- •11 Explicit CANCEL action
- •12 Feedback for completion of data entry
- •13 ▶ Feedback for repetitive data entries
- •14 ▶ Feedback when changing data
- •15 Keeping data items short
- •16 ▶ Partitioning long data items
- •17 Optional abbreviation
- •18 ▶ Distinctive abbreviation
- •19 ▶ Simple abbreviation rule
- •20 ▶ Minimal exceptions to abbreviation rule
- •21 ▶ Minimal deviation from abbreviation rule
- •22 ▶ Fixed abbreviation length
- •23 ▶ Clarifying unrecognized abbreviations
- •24 Prompting data entry
- •25 Character entry via single keystroke
- •26 ▶ Minimal shift keying
- •27 Upper and lower case equivalent
- •28 Decimal point optional
- •29 Leading zeros optional
- •30 Single and multiple blanks equivalent
- •31 Aids for entering hierarchic data
- •32 Speech input
- •33 ▶ Limited vocabulary for speech input
- •34 ▶ Phonetically distinct vocabulary for speech input
- •35 ▶ Easy error correction for speech input
- •36 ▶ Alternative entries for speech input
- •37 ▶ PAUSE and CONTINUE options for speech input

1.1 Position Designation

- •1 Distinctive cursor
- •2 ▶ Nonobscuring cursor
- •3 ▶ Precise pointing
- •4 Explicit activation
- •5 Fast acknowledgment of entry
- •6 Stable cursor
- •7 Responsive cursor control
- •8 Consistent incremental positioning
- •9 ▶ Variable step size
- •10 ▶ Proportional spacing
- •11 Continuous cursor positioning
- •12 Direct pointing
- •13 ▶ Large pointing area for option selection
- •14 Cursor control at keyboard
- •15 Compatible control of cursor movement
- •16 Minimal use of multiple cursors
- •17 ▶ Distinctive multiple cursors
- •18 ▶ Distinctive control of multiple cursors
- •19 ▶ Compatible control of multiple cursors
- •20 Consistent HOME position
- •21 Consistent cursor placement
- •22 Easy cursor movement to data fields
- •23 Display format protection
- •24 Data entry independent of cursor placement

1.2 Direction Designation

- •1 Analog entry of estimated direction
- •2 Keyed entry of quantified direction

DATA ENTRY

1.3 Text

- •1 Adequate display capacity
- •2 Editing capabilities during text entry
- •3 Free cursor movement
- •4 ▶ Control entries distinct from text
- •5 Natural units of text
- •6 ▶ Control entry based on units of text
- •7 ▶ Highlighting specified text
- •8 ▶ Cursor movement by units of text
- •9 String search
- •10 ▶ Upper and lower case equivalent in search
- •11 ▶ Specifying case in search
- •12 ▶ Global search and replace
- •13 ▶ Case in global search and replace
- •14 Automatic pagination aids
- •15 ▶ User control of pagination
- •16 ▶ Controlling integrity of text units
- •17 Automatic line break
- •18 ▶ Consistent word spacing
- •19 ▶ Hyphenation by users
- •20 Format control by user
- •21 Establishing predefined formats
- •22 ▶ Storing user-defined formats
- •23 Moving text
- •24 Storing frequently used text
- •25 Necessary data displayed
- •26 ▶ Text distinct from annotation
- •27 Text displayed as printed
- •28 Flexible printing options
- •29 Information on printing status
- •30 Auditory signals for alerting users
- •31 Protecting text during page overruns
- •32 Confirming actions in DELETE mode
- •33 Reversible actions
- •34 User confirmation of editing changes

1.4 Data Forms

- •1 Combined entry of related data
- •2 Flexible interrupt
- •3 Minimal use of delimiters
- •4 ▶ Standard delimiter character
- •5 Data field labels
- •6 ▶ Consistent labeling
- •7 ▶ Protected labels
- •8 ▶ Labels close to data fields
- •9 ▶ Standard symbol for prompting entry
- •10 Marking field boundaries
- •11 ▶ Prompting field length
- •12 ▶ Marking required and optional data fields
- •13 ▶ Field markers not entered with data
- •14 Automatic justification of variable-length entries
- •15 Explicit tabbing to data fields
- •16 Distinctive label format
- •17 ▶ Consistent label format
- •18 ▶ Label punctuation as entry cue
- •19 Informative labels
- •20 Data format cueing in labels
- •21 ▶ Labeling units of measurement
- •22 ▶ Familiar units of measurement
- •23 ▶ Alternative units of measurement
- •24 Form compatible for data entry and display
- •25 ▶ Form compatible with source documents
- •26 Minimal Cursor positioning
- •27 ▶ Data items in logical order
- •28 Automatic cursor placement

1.5 Tables

- •1 Tables for related data sets
- •2 Distinctive labels
- •3 ▶ Informative labels
- •4 Tabbing within rows
- •5 ▶ Tabbing within columns
- •6 Automatic justification of entries
- •7 ▶ Justification of numeric entries
- •8 ▶ Maintaining significant zeros
- •9 Aiding entry of duplicative data
- •10 Row scanning cues

DATA ENTRY

1.6 Graphics

- •1 Pointing
- •2 ▶ Distinctive cursor
- •3 ▶ Easy cursor positioning
- •4 Confirming cursor position
- •5 Zooming for precise positioning
- •6 Selecting graphic elements
- •7 ▶ Highlighting selected elements
- •8 Changing position (translation)
- •9 Deleting elements
- •10 Selecting from displayed attributes
- •11 ▶ Selecting colors
- •12 ▶ Displaying current attributes
- •13 Changing attributes
- •14 ▶ Consistent method for attribute selection
- •15 Easy storage and retrieval
- •16 ▶ Naming displays and elements
- •17 Automatic data registration
- •18 Aids for entering hierarchic data
- •19 Automatic data validation

1.6.1 Plotting Data

- •1 Automated data plotting
- •2 Plotting stored data
- •3 Predefined graphic formats
- •4 Aids for graph construction
- •5 ▶ Aids for scaling
- •6 Computer derivation of graphic data

1.6.2 Drawing

- •1 Drawing lines
- •2 Rubberbanding
- •3 Aiding line connection
- •4 Grid reference for alignment
- •5 ▶ Changing grid intervals
- •6 Constraint for vertical and horizontal lines
- •7 Specifying line relations
- •8 Drawing figures
- •9 ▶ Alternative methods for drawing figures
- •10 Changing size

1.6.2 Drawing (cont.)

- •11 ▶ Enlargement for symbol drawing
- •12 Copying elements
- •13 Rotating elements
- •14 Reflection of elements
- •15 Grouping elements
- •16 Merging elements
- •17 Filling enclosed areas
- •18 Automatic figure completion
- •19 Stored models

1.7 Data Validation

- •1 Automatic data validation
- •2 Accepting correct entries
- •3 Non-disruptive error messages
- •4 Deferral of required data entry
- •5 ▶ Reminder of deferred entry
- •6 Timely validation of sequential transactions
- •7 Optional item-by-item validation

1.8 Other Data Processing

- •1 Default values
- •2 ▶ Defaults for sequential entries
- •3 ▶ User definition of default values
- •4 ▶ Display of default values
- •5 ▶ Easy confirmation to enter default values
- •6 ▶ Temporary replacement of default values
- •7 Automatic generation of routine data
- •8 Automatic computation of derived data
- •9 User review of prior entries
- •10 Automatic entry of redundant data
- •11 Automatic cross-file updating
- •12 Aids for entering hierarchic data

1.9 Design Change

- •1 Flexible design for data entry

DATA DISPLAY

2.0 General

- •1 Necessary data displayed
- •2 ▶ Only necessary data displayed
- •3 Data displayed in usable form
- •4 Data display consistent with user conventions
- •5 ▶ Establishing display standards
- •6 Consistent display format
- •7 Display consistent with entry requirements
- •8 User control of data display
- •9 ▶ User changes to displayed data
- •10 ▶ Protection of displayed data
- •11 Context for displayed data
- •12 Familiar wording
- •13 ▶ Consistent wording
- •14 ▶ Consistent wording across displays
- •15 ▶ Consistent grammatical structure
- •16 Minimal use of abbreviation
- •17 ▶ Common abbreviations
- •18 ▶ Simple abbreviation rule
- •19 ▶ Distinctive abbreviations
- •20 ▶ Minimal punctuation of abbreviations
- •21 ▶ Dictionary of abbreviations

2.1 Text

- •1 Conventional text display
- •2 Printing lengthy text displays
- •3 Consistent text format
- •4 ▶ Adequate display capacity
- •5 ▶ Text displayed in wide columns
- •6 ▶ Conventional use of mixed case
- •7 ▶ Separation of paragraphs
- •8 ▶ Consistent word spacing
- •9 ▶ Minimal hyphenation
- •10 ▶ Conventional punctuation
- •11 Clarity of wording
- •12 ▶ Sentences begin with main topic
- •13 ▶ Simple sentence structure
- •14 ▶ Concise wording
- •15 ▶ Distinct wording
- •16 ▶ Affirmative sentences
- •17 ▶ Active voice
- •18 ▶ Temporal sequence
- •19 Lists for related items
- •20 ▶ Single-column list format

2.1 Text (cont.)

- •21 ▶ Marking multiline items in a list
- •22 ▶ Arabic numerals for numbered items
- •23 ▶ Logical list ordering
- •24 ▶ Vertical ordering in multiple columns
- •25 ▶ Hierarchic structure for long lists
- •26 Abbreviations defined in text
- •27 Highlighting text
- •28 Combining text with other data
- •29 ▶ Placing figures near their citations

2.2 Data Forms

- •1 Forms for related data
- •2 Visually distinctive data fields
- •3 ▶ Data field labeling
- •4 ▶ Descriptive wording of labels
- •5 ▶ Consistent wording of labels
- •6 ▶ Distinctive wording of labels
- •7 ▶ Consistent label location
- •8 ▶ Distinctive label format
- •9 ▶ Labels close to data fields
- •10 ▶ Labeling units of measurement
- •11 Consistent format across displays
- •12 ▶ Form compatible for data entry and display
- •13 Consistent format within data fields
- •14 Partitioning long data items
- •15 Distinguishing blanks from nulls

DATA DISPLAY

2.3 Tables

•1 Tables for data comparison
•2 Logical organization
•3 Table access by row and column
•4 ▶ Tables referenced by first column
•5 ▶ Items paired for direct comparison
•6 Row and column labels
•7 ▶ Consistent label format
•8 ▶ Distinctive labeling
•9 ▶ Numbered items start with "1"
•10 ▶ Repeated elements in hierarchic numbering
•11 ▶ Labeling units of measurement
•12 Consistent column spacing
•13 ▶ Column scanning cues
•14 ▶ Row scanning cues
•15 Justification of alphabetic data
•16 Justification of numeric data
•17 ▶ Maintaining significant zeros

2.4 Graphics

•1 Graphic displays
•2 ▶ Data comparison
•3 ▶ Monitoring data change
•4 Consistency
•5 Only necessary information displayed
•6 Highlighting critical data
•7 Reference index or baseline
•8 Text annotation
•9 ▶ Data annotation
•10 ▶ Consistent annotation format
•11 ▶ Normal orientation for labels
•12 Standard symbols
•13 ▶ Pictorial symbols
•14 Simple texture codes
•15 Zooming for display expansion
•16 ▶ Show changing scale
•17 ▶ Show overview position of visible section
•18 Animation for dynamic display
•19 ▶ Highlighting by animation
•20 Printing graphic displays

2.4.1 Scaling

•1 Scaling conventions
•2 ▶ Consistent scaling
•3 Labeling axes
•4 Linear scaling
•5 ▶ Scaling in standard intervals
•6 ▶ Numeric scales start at zero
•7 Restricted use of broken axes
•8 Duplicate axes
•9 Single scale only
•10 ▶ Scaling against a reference index
•11 Aids for scale interpolation
•12 ▶ Unobtrusive grids
•13 Restricted use of three-dimensional scaling

2.4.2 Scatterplots

•1 Scatterplots
•2 Highlighting
•3 Grouping scatterplots to show multiple relations
•4 ▶ Interactive analysis of grouped scatterplots

2.4.3 Curves and Line Graphs

•1 Curves and line graphs
•2 Comparing curves
•3 Labeling curves
•4 ▶ Compatible ordering in legends
•5 Highlighting
•6 Line coding to distinguish curves
•7 ▶ Consistent line codes
•8 ▶ Broken lines for projected curves
•9 Reference index
•10 Repeating display of cyclic data
•11 Direct display of differences
•12 Surface charts
•13 ▶ Ordering data in surface charts
•14 ▶ Labeling surface charts
•15 Cumulative curves

DATA DISPLAY

2.4.4 Bar Graphs

- •1 Bar graphs
- •2 ▶ Histograms (step charts)
- •3 Consistent orientation of bars
- •4 Bar spacing
- •5 Reference index
- •6 Highlighting
- •7 Paired or overlapped bars
- •8 ▶ Labeling paired bars
- •9 Stacked or segmented bars
- •10 ▶ Ordering data in stacked bars
- •11 Restricted use of icons

2.4.5 Pie Charts

- •1 Restricted use of pie charts
- •2 Labeling pie charts
- •3 ▶ Numeric labels
- •4 Highlighting

2.4.6 Pictures and Diagrams

- •1 Pictures
- •2 Diagrams
- •3 Linking sectional diagrams
- •4 Highlighting
- •5 Rotation
- •6 Aids for pictorial analysis

2.4.7 Flowcharts

- •1 Flowcharts
- •2 ▶ Flowcharts to aid problem solving
- •3 Logical ordering of steps
- •4 ▶ Ordering to minimize path length
- •5 ▶ Conventional path orientation
- •6 Consistent coding of elements
- •7 ▶ Conventional use of arrows
- •8 Highlighting
- •9 Single decision at each step
- •10 Logical ordering of options
- •11 ▶ Consistent ordering of options
- •12 ▶ Consistent wording

2.4.8 Maps and Situation Displays

- •1 Maps
- •2 Consistent orientation
- •3 Consistent projection
- •4 Labeling maps
- •5 ▶ Consistent positioning of labels
- •6 Area coding
- •7 ▶ Tonal codes
- •8 ▶ Ordered coding
- •9 Highlighting
- •10 Panning for flexible display framing
- •11 ▶ Show overview position of visible section
- •12 Aiding distance judgments
- •13 Aids for analyzing maps
- •14 Mapping nongeographic data
- •15 Situation displays
- •16 Indicating data change
- •17 Selectable data categories
- •18 Stable reference for changing data

2.5 Format

- •1 Consistent format
- •2 Distinctive display elements
- •3 ▶ Spacing to structure displays
- •4 Paging crowded displays
- •5 ▶ Related data on same page
- •6 ▶ Page labeling
- •7 Integrated display
- •8 User-defined data windows
- •9 ▶ Adequate window size
- •10 Display title at top
- •11 Command entry, prompts, messages at bottom
- •12 Logical data oganization
- •13 ▶ Grouping for data comparison
- •14 ▶ Data grouped by sequence of use
- •15 ▶ Data grouped by function
- •16 ▶ Data grouped by importance
- •17 ▶ Data grouped by frequency
- •18 ▶ Data grouped alphabetically or chronologically

DATA DISPLAY

DATA DISPLAY

2.7.3 Update

- •1 Automatic display update
- •2 Highlighting changed data
- •3 Readability of changing data
- •4 Visual integration of changing graphics
- •5 Display freeze
- •6 ▶ Labeling display freeze
- •7 ▶ Signaling changes to frozen data
- •8 ▶ Resuming update after display freeze
- •9 Prediction display

2.7.4 Suppression

- •1 Temporary suppression of displayed data
- •2 ▶ Labeling display suppression
- •3 ▶ Signaling changes to suppressed data
- •4 ▶ Resuming display of suppressed data

2.7.5 Window Overlays

- •1 Temporary window overlays
- •2 Predefined windows
- •3 User-specified windows
- •4 Consistent window control
- •5 Easy suppression of window overlays
- •6 Labeling windows
- •7 Indicate active window
- •8 ▶ Easy shifting among windows
- •9 Consistent control within windows
- •10 Nondestructive overlay

2.8 Design Change

- •1 Flexible design for data display

SEQUENCE CONTROL

3.0 General

- •1 Flexible sequence control
- •2 Minimal user actions
- •3 Control matched to user skill
- •4 User initiative in sequence control
- •5 Control by explicit user action
- •6 Consistent user actions
- •7 Logical transaction sequences
- •8 Distinctive display of control information
- •9 Displayed context
- •10 Consistent terminology for sequence control
- •11 ▶ Congruent names for control functions
- •12 ▶ Upper and lower case equivalent
- •13 ▶ Wording consistent with user guidance
- •14 Feedback for control entries
- •15 ▶ Indicating completion of processing
- •16 ▶ Compatibility with user expectations
- •17 User-paced sequence control
- •18 Appropriate computer response time
- •19 Control availability
- •20 ▶ Indicating control lockout
- •21 ▶ Interrupt to end control lockout
- •22 Control by simultaneous users

3.1 Dialogue Type

- •1 Dialogue matched to task and user
- •2 Appropriate computer response time

3.1.1 Question and Answer

- •1 Question-and-answer dialogue
- •2 Questions displayed singly
- •3 Recapitulating prior answers
- •4 Sequence compatible with source documents

3.1.2 Form Filling

- •1 Form filling for data entry
- •2 Form filling for control entry
- •3 Defaults for control entry
- •4 Consistent format for control forms

3.1.3 Menu Selection

- •1 Menu selection
- •2 Single selection per menu
- •3 Single-column list format
- •4 Menu selection by pointing
- •5 ▶ Large pointing area for option selection
- •6 ▶ Dual activation for pointing
- •7 Menu selection by keyed entry
- •8 ▶ Standard area for code entry
- •9 Feedback for menu selection
- •10 Explanatory title for menu
- •11 Menu options worded as commands
- •12 ▶ Option wording consistent with command language
- •13 Letter codes for menu selection
- •14 ▶ Consistent coding of menu options
- •15 ▶ Standard symbol for prompting entry
- •16 Explicit option display
- •17 ▶ Complete display of menu options
- •18 ▶ Menu options dependent on context
- •19 Consistent display of menu options
- •20 Menus distinct from other displayed information
- •21 Logical ordering of menu options
- •22 Logical grouping of menu options
- •23 ▶ Logical ordering of grouped options
- •24 ▶ Labeling grouped options
- •25 Hierarchic menus for sequential selection
- •26 ▶ General menu
- •27 ▶ Minimal steps in sequential menu selection
- •28 ▶ Easy selection of important options
- •29 ▶ Automatic cursor placement
- •30 Indicating current position in menu structure
- •31 ▶ Control options distinct from menu branching
- •32 ▶ Consistent design of hierarchic menus
- •33 ▶ Return to higher-level menus
- •34 ▶ Return to general menu
- •35 By-passing menu selection with command entry
- •36 ▶ Stacking menu selections

SEQUENCE CONTROL

3.1.4 Function Keys

- •1 Function keys for critical control entries
- •2 ▶ Function keys for frequent control entries
- •3 ▶ Function keys for interim control entries
- •4 Distinctive labeling of function keys
- •5 ▶ Labeling multifunction keys
- •6 Single keying for frequent functions
- •7 ▶ Logical pairing of double-keyed functions
- •8 ▶ Consistent logic for double keying
- •9 Single activation of function keys
- •10 Feedback for function key activation
- •11 Indicating active function keys
- •12 ▶ Disabling unneeded function keys
- •13 Single key for continuous functions
- •14 Consistent assignment of function keys
- •15 ▶ Consistent functions in different operational modes
- •16 Easy return to base-level functions
- •17 Distinctive location
- •18 ▶ Layout compatible with use

3.1.5 Command Language

- •1 Command language
- •2 Standard display area for command entry
- •3 Functional command wording
- •4 Layered command language
- •5 Meaningful command names
- •6 ▶ Familiar wording
- •7 ▶ Consistent wording of commands
- •8 ▶ Distinctive meaning for commands
- •9 ▶ Distinctive spelling for commands
- •10 User-assigned command names

3.1.5 Command Language (cont.)

- •11 User-requested prompts
- •12 ▶ General list of commands
- •13 Command stacking
- •14 ▶ User definition of macro commands
- •15 Minimal punctuation
- •16 ▶ Standard delimiter
- •17 Ignoring blanks in command entry
- •18 Abbreviation of commands
- •19 Standard techniques for command editing
- •20 ▶ Interpreting misspelled commands
- •21 ▶ Recognizing command synonyms
- •22 ▶ Recognizing alternative syntax
- •23 Correcting command entry errors
- •24 ▶ Replacing erroneous commands
- •25 Reviewing destructive commands

3.1.6 Query Language

- •1 Query language
- •2 Natural organization of data
- •3 ▶ Coherent representation of data organization
- •4 Task-oriented wording
- •5 Flexible query formulation
- •6 Minimal need for quantifiers
- •7 Logic to link queries
- •8 Linking sequential queries
- •9 Confirming large-scale retrieval

3.1.7 Natural Language

- •1 Constrained natural language

3.1.8 Graphic Interaction

- •1 Control of graphic data
- •2 Graphic control aids for casual users
- •3 Iconic menus
- •4 ▶ Supplementary verbal labels
- •5 Direct manipulation
- •6 Graphic display of control context
- •7 Graphic display of control guidance

SEQUENCE CONTROL

3.2 Transaction Selection

- •1 User control in transaction selection
- •2 General list of control options
- •3 ▶ Organization and labeling of listed options
- •4 Indicating appropriate control options
- •5 ▶ Prompting control entries
- •6 Cursor placement for pointing at options
- •7 ▶ Cursor placement for keyed entry of options
- •8 Displaying option codes
- •9 Task-oriented wording for options
- •10 Only available options offered
- •11 Indicating control defaults
- •12 Consistent CONTINUE option
- •13 Stacked control entries
- •14 ▶ Consistent order in entry stacking
- •15 ▶ Abbreviation in entry stacking
- •16 ▶ Minimal punctuation of stacked entries
- •17 ▶ Standard delimiter in entry stacking
- •18 User definition of macro commands
- •19 User-specified transaction timing

3.3 Interrupt

- •1 User interruption of transactions
- •2 Distinctive interrupt options
- •3 CANCEL option
- •4 BACKUP option
- •5 REVIEW option
- •6 RESTART option
- •7 END option
- •8 PAUSE and CONTINUE options
- •9 ▶ Indicating PAUSE status
- •10 SUSPEND option
- •11 ▶ Indicating SUSPEND status

3.4 Context Definition

- •1 Defining context for users
- •2 ▶ Context established by prior entries
- •3 ▶ Record of prior entries
- •4 Display of operational mode
- •5 Display of control parameters
- •6 Highlighting selected data
- •7 Consistent display of context information

3.5 Error Management

- •1 Appropriate response to all entries
- •2 Command editing
- •3 Prompting command correction
- •4 Errors in stacked commands
- •5 ▶ Partial execution of stacked commands
- •6 Explicit entry of corrections
- •7 User confirmation of destructive entries
- •8 ▶ User warned of potential data loss
- •9 ▶ Distinctive CONFIRM action
- •10 UNDO to reverse control actions
- •11 ▶ Preventing data loss at LOG-OFF
- •12 ▶ Immediate data correction
- •13 ▶ Flexible BACKUP for error correction

3.6 Alarms

- •1 Alarm definition by user
- •2 Distinctive and consistent alarms
- •3 Alarm acknowledgement
- •4 ▶ Alarm reset
- •5 ▶ Special acknowledgement of critical alarms

3.7 Design Change

- •1 Flexible design for sequence control

USER GUIDANCE

4.0 General

- •1 Standard procedures
- •2 Explicit user actions
- •3 Separate LOG-ON procedure
- •4 Display of guidance information
- •5 Only necessary information displayed
- •6 Consistent display format
- •7 ▶ Consistent format for user guidance
- •8 Distinctive format for user guidance
- •9 ▶ Distinctive cursor
- •10 Clear control labels
- •11 Clear data labels
- •12 Highlighting critical user guidance
- •13 Consistent coding conventions
- •14 ▶ Familiar coding conventions
- •15 Consistent wording
- •16 ▶ Familiar wording
- •17 ▶ Task-oriented wording
- •18 ▶ Wording consistent with control entry
- •19 ▶ Speaking directly to users
- •20 ▶ Affirmative statements
- •21 ▶ Active voice
- •22 ▶ Temporal sequence
- •23 ▶ Consistent grammatical structure
- •24 Flexible user guidance
- •25 ▶ Easy ways to get guidance
- •26 Speech output
- •27 ▶ Limited number of spoken messages
- •28 ▶ Simple spoken messages
- •29 ▶ Distinctive spoken warnings

4.1 Status Information

- •1 Indicating status
- •2 Automatic LOG-ON display
- •3 ▶ LOG-ON delay
- •4 Keyboard lock
- •5 Operational mode
- •6 Other users
- •7 System load
- •8 External systems
- •9 Date and time signals
- •10 Alarm settings

4.2 Routine Feedback

- •1 Consistent feedback
- •2 Fast response
- •3 Feedback for control entries
- •4 ▶ Indicating completion of processing
- •5 Feedback for print requests
- •6 Display identification
- •7 ▶ Identifying multipage displays
- •8 Indicating operational mode
- •9 ▶ Indicating option selection
- •10 ▶ Indicating item selection
- •11 Feedback for user interrupt

4.3 Error Feedback

- •1 Informative error messages
- •2 ▶ Specific error messages
- •3 ▶ Task-oriented error messages
- •4 Advisory error messages
- •5 Brief error messages
- •6 Neutral wording for error messages
- •7 Multilevel error messages
- •8 Multiple error messages
- •9 ▶ Indicating repeated errors
- •10 Non-disruptive error messages
- •11 Appropriate response time for error messages
- •12 Documenting error messages
- •13 Cursor placement following error
- •14 Displaying erroneous entries
- •15 User editing of entry errors
- •16 Removing error messages
- •17 Cautionary messages
- •18 User confirmation of destructive entries
- •19 Alarm coding

USER GUIDANCE

4.4 Job Aids

- •1 Guidance information always available
- •2 General list of control options
- •3 Logical menu structure
- •4 Hierarchic menus
- •5 Guidance for sequence control
- •6 ▶ Transaction specific option display
- •7 Prompting entries
- •8 ▶ Standard display location for prompting
- •9 ▶ Consistent format for prompts
- •10 ▶ Standard symbol for prompting entry
- •11 ▶ Concise wording of prompts
- •12 ▶ User-requested prompts
- •13 Displayed context
- •14 ▶ Maintaining context for data entry
- •15 Cues for prompting data entry
- •16 Consistent cursor positioning
- •17 On-line system guidance
- •18 ▶ Index of data
- •19 ▶ Index of commands
- •20 ▶ Dictionary of abbreviations
- •21 Definition of display codes
- •22 Record of past transactions
- •23 HELP
- •24 ▶ Standard action to request HELP
- •25 ▶ Task-oriented HELP
- •26 ▶ Clarifying HELP requests
- •27 ▶ Synonyms for standard terminology
- •28 ▶ Multilevel HELP
- •29 ▶ Browsing HELP
- •30 On-line training
- •31 ▶ Flexible training
- •32 ▶ Adaptive training

4.5 User Records

- •1 User performance measurement
- •2 Notifying users
- •3 Transaction records
- •4 Data access records
- •5 Records of program use
- •6 Error records
- •7 HELP records

4.6 Design Change

- •1 Flexible design for user guidance
- •2 Notifying users of design changes

DATA TRANSMISSION

5.0 General

•1 Functional integration
•2 Functional wording
•3 Consistent procedures
•4 Minimal memory load on user
•5 Minimal user actions
•6 Control by explicit user action
•7 Flexible user control
•8 ▶ Interrupt
•9 Flexible message filing
•10 Message highlighting

5.1 Preparing Messages

•1 Message composition compatible with data entry
•2 User-designed message formats
•3 ▶ Unformatted text
•4 Stored message forms
•5 Automatic message formatting
•6 ▶ Automatic text formatting
•7 ▶ Data forms
•8 ▶ Tables and graphics
•9 Flexible data specification
•10 ▶ Incorporate existing files
•11 ▶ Incorporate other messages
•12 Variable message length
•13 Saving draft messages

5.2 Addressing Messages

•1 Destination selection
•2 Standard address header
•3 Prompting address entry
•4 Address directory
•5 ▶ Aids for directory search
•6 ▶ Extracting directory addresses
•7 User-assigned nicknames for addressing
•8 System distribution lists
•9 ▶ Access to distribution list information
•10 Informal distribution lists

5.2 Addressing Messages (cont.)

•11 ▶ Lists within lists
•12 ▶ Modifying distribution lists
•13 Automatic expansion of partial addresses
•14 Automatic address checking
•15 Addressing replies to messages received
•16 Editing address headers
•17 Single occurrence of address
•18 Serial distribution
•19 Redistributing received messages

5.3 Initiating Transmission

•1 User-initiated transmission
•2 User review before transmission
•3 ▶ Optional message display
•4 Computer-initiated transmission
•5 Information about communication status
•6 Sender identification
•7 Assignment of priority
•8 Automatic queuing for transmission
•9 Deferring message transmission
•10 ▶ Transmission at specified date/time
•11 Return receipt
•12 Cancel transmission
•13 Printing messages

5.4 Controlling Transmission

•1 Automatic protection of transmitted data
•2 Automatic feedback
•3 ▶ User specification of feedback
•4 Send single copy
•5 Queuing failed transmissions
•6 ▶ Saving undelivered messages
•7 ▶ Notification of transmission failure
•8 Message recall
•9 Automatic record keeping

DATA TRANSMISSION

5.5 Receiving Messages

•1 Specifying sources
•2 Specifying device destination
•3 Queuing messages received
•4 Message notification at LOG-ON
•5 Nondisruptive notification of arriving messages
•6 Indicating priority of received messages
•7 Filters for message notification
•8 Warning of incompatible format
•9 Information about queued messages
•10 ▶ Indicate message size
•11 ▶ Specifying format for message listings
•12 User review of messages in queue
•13 Filters for ordering message review
•14 Message review compatible with data display
•15 Labeling received messages
•16 Annotating received messages
•17 Filters for message filing
•18 Discarding messages

5.6 Design Change

•1 Flexible design for data transmission

DATA PROTECTION

6.0 General

•1 Automated security measures
•2 Warning of threats to security
•3 Protection from computer failure
•4 Protection from interference by other users
•5 Protection from interrupts
•6 Segregating real from simulated data
•7 Consistent procedures
•8 Appropriate ease or difficulty of user actions
•9 Control by explicit user action
•10 User review and editing of entries
•11 Disabling unneeded controls
•12 ▶ Protecting physical controls
•13 Safe defaults
•14 Safe response to random inputs
•15 Explicit action to select destructive modes
•16 Feedback for mode selection
•17 Warning users of potential data loss
•18 User confirmation of destructive actions
•19 ▶ Distinctive CONFIRM action
•20 ▶ Separate CONFIRM action
•21 Reversible control actions (UNDO)

6.1 User Identification

•1 Easy LOG-ON
•2 ▶ Prompting LOG-ON
•3 User choice of passwords
•4 ▶ Changing passwords
•5 ▶ Private entry of passwords
•6 Limiting unsuccessful LOG-ON attempts
•7 Auxiliary tests to authenticate user identity
•8 Continuous recognition of user identity

6.2 Data Access

•1 Single authorization for data access
•2 Displayed security classification
•3 Protecting displayed data
•4 ▶ Indicating "read-only" displays
•5 Protecting display formats
•6 Display suppression for security
•7 Protecting printed data
•8 Automatic records of data access
•9 Encryption
•10 ▶ Ensuring reversible encryption

6.3 Data Entry/Change

•1 Single authorization for data entry/change
•2 Protection from data change
•3 Data entry/change transaction records
•4 Simple procedures
•5 Explicit user actions
•6 ▶ Single entry of related data
•7 ▶ Data entry independent of cursor placement
•8 Editing data before entry
•9 Immediate error correction
•10 ▶ Editing entries after error detection
•11 ▶ Explicit entry of corrections
•12 Flexible BACKUP for error correction
•13 Data verification by user review
•14 ▶ Automatic data generation
•15 ▶ Displaying default values
•16 Displaying data to be changed
•17 Validating data changes
•18 ▶ Cross validation of related data
•19 User confirmation of destructive actions
•20 Distinctive file names
•21 Segregating real from simulated data
•22 Preventing data loss at LOG-OFF

DATA PROTECTION

6.4 Data Transmission

- •1 Automatic protection of transmitted data
- •2 User review of data before transmission
- •3 Encrypting messages
- •4 Saving sent data until receipt is confirmed
- •5 Nondisruptive notification of messages received
- •6 Authenticating message sources
- •7 Printing messages

6.5 Design Change

- •1 Flexible design for data protection
- •2 Protection from design change

GLOSSARY

A comprehensive glossary of words dealing with the user interface to computer systems could contain hundreds of terms. Such a glossary would be difficult to compile, because terms are often used inconsistently. The glossary presented here is not intended to be a comprehensive reference. Instead, it simply attempts to clarify the meanings of some of the terms used in this report. A term may be included in this glossary if it is used inconsistently in the general literature, or perhaps if its meaning in this report is more narrow than its commonly understood meaning.

Addressing Messages - Preparing header information to specify the destination for data to be transmitted.

Attribute - A characteristic of a displayed element such as color, bolding, size, pattern or font, which can be specified by a user.

BACKUP - A capability that returns a user to the last previous display in a defined transaction sequence.

Bar graph - A graphic figure in which numeric quantities are represented by the linear extent of parallel lines (or bars). Bar graphs are useful for showing a comparative measure for separate entities or for a variable sampled at discrete intervals.

CANCEL - A capability that regenerates (or re-initializes) the current display without processing or retaining any changes made by the user.

Category - A grouping of data values along a dimension defined for operational purposes. For example, an air traffic controller might wish to implement the same procedures for all aircraft with speeds in the category of 600 to 800 knots. See also "Value".

Command Language - A type of dialogue in which a user composes control entries, possibly with prompting by the computer.

Context Definition - Displaying an indication of previous user actions or computer processing that will affect the results of current actions, in order to help a user predict how the system will respond.

Control Entry - User input for sequence control, such as function key activation, menu selection, command entry, etc.

Controlling Transmission - Ensuring that transmitted data are delivered, saved until they can be delivered, or returned to the sender. A computer will usually control tranmission, but users may need information about that process.

Cursor - A marker on the display screen that indicates the current position for attention, which may designate a displayed item. A cursor might be positioned under computer control or by the user.

Curve - A graphed line that shows the relation between sets of data defined by two continuous variables. On a curve, data points are sufficiently close to appear as a smooth line.

Data - The raw materials from which a user extracts information. Data may include numbers, words, pictures, etc.

Data Base - A collection of data that is stored in the computer.

Data Display - Output of data from a computer to its users. Generally, this phrase denotes visual output, but it may be qualified to indicate a different modality, such as an "auditory display".

Data Entry - User input of data for computer processing, and computer responses to such inputs.

Data Field - An area of the display screen reserved for user entry of a data item.

Data Field Label - A displayed word or phrase that serves as a prompt for entering an item in a data field. Such a label usually cannot be changed by a user.

Data Item - A set of characters of fixed or variable length that forms a single unit of data. Examples of a data item might be a person's last name, or a ZIP code. Sometimes a data item might contain only a single character. Data items may be entered by a user or may be supplied by the computer.

Data Protection - Functional capabilities that guard against unauthorized data access and tampering, and data loss due to user errors or computer failure.

Data Transmission - Computer-mediated communication among system users, and also with other systems. Transmitted data may include numbers, words, pictures, etc.

Data Validation - Functional capabilities that check data entry items for correct content or format, as defined by software logic.

Default Value - A predetermined, frequently used, value for a data or control entry, intended to reduce required user entry actions.

Diagram - A special form of a picture in which details are only shown if they are necessary for the performance of a task. For example, an electrical wiring diagram for a building would show wiring but not necessarily furniture or plumbing.

Dialogue - A structured series of interchanges between a user and a computer. A dialogue can be initiated by a computer (e.g., question and answer) or by a user (e.g., command language).

Dimension - A scale or categorization along which data may vary, taking different values at different times. For example, relevant dimensions for an aircraft might include its heading, speed and altitude. See also "Variable".

Direction Designation - User entry of directional data (azimuth, bearing, heading) on a display.

Display - See "Data Display".

Display Control - Procedures by which a user can specify what data are shown, and how.

Display Format - The organization of different types of data in a display, including information about the data such as labels, and other user guidance such as prompts, error messages, etc.

Display Framing - User control of data coverage by display movement, including paging, scrolling, offset, expansion, etc.

Display Selection - Refers to the specification of data outputs, either by a user or automatically.

Display Tailoring - Designing displays to meet the specific task needs of a user, rather than providing a general display which can be used for many purposes.

Display Update - Regeneration of changing data to show current status, by user request or automatically by the computer.

Drawing - Using computer aids to specify lines and other graphic elements, in order to create a pictorial representation.

ENTER - An explicit user action that effects computer processing of user entries. For example, after typing a series of numbers, a user might press an ENTER key that will add them to a data base, subject to data validation.

Entry - See "Data Entry" or "Control Entry".

Error Management - Interface design to facilitate the detection and correction of user errors.

Field - See "Data Field".

File - A collection of data, treated as a single unit, that is stored in the computer.

Flowchart - A diagram that illustrates sequential relations among elements or events. Flowcharts are often shown as boxes connected by arrows.

Format - See "Display Format".

Form Filling - A type of dialogue in which the computer displays forms containing labeled fields for data entry by a user.

Framing - See "Display Framing".

Function - A computer-supported capability provided to users as an aid for task performance. Examples of functions are position designation or direction designation.

Function Key - A key whose activation will effect a control entry.

Graphics - Data specially formatted to show spatial, temporal, or other relations among data sets.

Graphic Element - A component part of a graphic display, such as a line, a circle, or a scale.

Graphic Interaction - A kind of dialogue which permits a user to select displayed control elements by pointing and other direct manipulation.

Hard Copy - A printed paper display output by the computer.

HELP - A capability that displays information upon user request for on-line guidance. HELP may inform a user generally about system capabilities, or may provide more specific guidance in information handling transactions.

Highlighting - Emphasizing displayed data or format features in some way, e.g., through the use of underlining, bolding, or inverse video.

Information - Organized data that users need to successfully perform their tasks. Information serves as an answer to a user's question about data. It is used here to refer to the effective assimilation of data by a user.

Information System - A computer-supported, task-oriented tool designed to help users perform defined information handling tasks.

Initiating Transmission - The process of actually sending a prepared message or data file. Transmission can either be initiated by the computer, or by a system user.

Input - See "Control Entry" and "Data Entry".

Interaction - See "Transaction".

Interface - See "User-System Interface".

Interrupt - Stopping an ongoing transaction in order to redirect the course of the processing. Examples of interrupt options are BACKUP, REVIEW, CANCEL, RESTART.

Item - See "Data Item".

Job - A set of responsibilites and activities that are defined for each system user.

Label - A title or descriptor that helps a user identify displayed data. See "Data Field Label".

Line Graph - A scaled figure that shows relations among sets of data defined by two continuous variables. On a line graph, data points are connected by straight line segments.

Menu Selection - A type of dialogue in which a user selects one item out of a list of displayed alternatives, whether the selection is by pointing, by entry of an associated option code, or by activation of an adjacent function key.

Message - Refers specifically to data that are transmitted as a discrete transaction from one computer user to another along with formal header information, and may sometimes refer loosely to other forms of data transmission. See "Data Transmission".

Natural Language - A type of dialogue in which a user composes control entries in natural language (e.g., English, Spanish, French) rather than by a more formally constrained command language.

Operator - See "User".

Output - See "Data Display".

Page - The data appearing at one time on a single display screen.

Panning - An orientation for display framing in which a user conceives of the display frame as moving over a fixed array of data. The opposite of scrolling.

Picture - A detailed representation of a real or imaginary object or event.

Pie Charts - A circle divided into sections (as pieces of a pie) in order to represent graphically the relative proportions of different parts of a whole.

Plotting Data - Locating data points on a scaled graphic display by means of coordinates.

Position Designation - User selection and entry of a position on a display, or of a displayed item. See also "Cursor".

Preparing Messages - Includes specification of contents, format and header information.

Query Language - A type of dialogue in which a user composes control entries requesting specified data from a data base.

Question and Answer - A type of dialogue in which a computer displays questions, one at a time, for a user to answer.

Receiving Messages - Includes provision of computer aids for queuing, reviewing, filing or otherwise disposing of data transmitted to a user from some other source.

RESTART - A capability that cancels all user entries in a defined transaction sequence.

REVIEW - A capability that returns a user to the first display in a defined transaction sequence, while retaining any entries made by the user.

Scaling - The positioning of displayed data elements with respect to a defined measurement standard.

Scatterplot - A scaled graph which shows relations among individual data points in a two dimensional array.

Screen - See "Page".

Scrolling - An orientation for display framing in which a user conceives of data as moving behind a fixed display frame. The opposite of panning.

Selecting - A user's action of identifying display elements to the computer in order to manipulate those elements in some way, e.g., to move them, or change their attribute(s), or delete them.

Selection, Display - See "Display Selection".

Sequence Control - Logic and means by which user actions and computer responses are linked to become coherent transactions.

Situation Display - A type of diagram on which changing data are shown on a fixed background.

Specification - See "Selection".

Status Information - Information on current data processing status which is displayed to a user either automatically or by user request, perhaps indicating system load, keyboard lock, or processing delay.

Suppression - User control of displayed data by temporary deletion of specified data categories.

Suspense File - A temporary collection of data saved by a computer for later use.

System - See "Information System".

Task - A series of transactions that comprises part of a user's defined job.

Terminal - An input/output device used to enter and display data. Data are usually entered via a keyboard, and are usually displayed via a video screen ("soft copy") or a printer ("hard copy").

Text Entry - Initial entry and subsequent editing of textual data.

Transaction - An action by a user followed by a response from the computer. Transaction is used here to represent the smallest functional unit of user-system interaction.

Transmission Control - Controlling the sequence, content, format, routine, timing, etc., of data transmission.

Update - See "Display Update".

User - Any person who uses an information system in performing his/her job.

User Guidance - Computer prompts and feedback that aid users in performing their tasks. Examples include data field labels, alarm or alert signals, error messages, and HELP displays.

User Records - Automatic recording of user performance, e.g., data access records, error records, requests for HELP.

User-System Interface - All aspects of information system design that affect a user's participation in information handling transactions.

Value - Specific data for a particular dimension or variable. For example, values for an aircraft's speed might be 800 knots during one observation and 500 knots during another. See also "Category".

Variable - See "Dimension".

Window - A portion of a display screen showing a particular kind of information, e.g., a command entry window, or a window for displaying error messages. Windows may be stable features of a display format, i.e, defined by system designers, or may be temporary, i.e., window overlays requested by a user for temporary display of data, menus, etc.

Window Overlay - A portion of a display that is temporarily used to show added features such as requested data, menus, user guidance, etc., which may obscure initially-displayed data.

Work Station - The physical facilities with which a user works and the relevant environment, including such things as computer terminals, source documents, desks, chairs, and lighting.

INDEX

A

E

F

S

T

X

Y

Z